Report No. 404

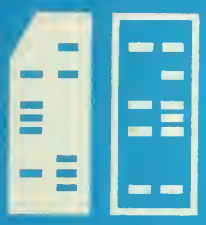
COO-1469-0167

MULTIPLE PRECISION COMPUTATION OF LEGENDRE FUNCTIONS

by

Ruth Ann Potts Fleck

June, 1970



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

MULTIPLE PRECISION COMPUTATION OF LEGENDRE FUNCTIONS

BY

RUTH ANN POTTS FLECK
A.B., Ripon College, 1966

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois, 1970

Urbana, Illinois

ACKNOWLEDGMENT

The author wishes to thank Professor L. D. Fosdick for his supervision and guidance in the preparation of this thesis.

Also sincere appreciation is extended to this author's husband for his patience, assistance and encouragement during the writing of this paper.

The computational work described in this thesis was performed on the I.B.M. 360 75 at the University of Illinois. This research was supported by TR US AEC Contract 1469.

TABLE OF CONTENTS

I. INTRODUCTION

In the solution of problems arising in applied mathematics, it is often necessary to know the value of certain functions. Many of these functions are difficult or extremely tedious and time consuming to evaluate and so it has become traditional for tables of often used functions to be published. The increased use and efficiency of the digital computer has both solved and created many problems in this area of function evaluation. It is now possible to create tables much faster than by human calculation and consequently to tabulate functions for many more values of the argument and to a greater number of decimal places. However the use of these tables is now more difficult. A physicist writing a computer program to solve a problem does not want to stop the program at the point where he needs a function value, print out the argument, go to a table to look up the function value, read it into the computer and resume running the program. It is impractical to have the tables themselves stored in the computer because of the storage space consumed and to have the tables on punched cards implies large volumes of physical storage space for the cards plus time to read the cards each time a program is run. A good solution would seem to be storage of tables on magnetic tape which requires less physical room and is much faster on input, in fact the whole table would not have to be read in every time if there were section markers on the tape. Many decisions would have to be made to make this system workable in a universal sense. Perhaps the most difficult is how to store the numbers

on tape so they can be read by many different computers. Another important consideration is for what values of the argument should the function be evaluated so that the table is adequate, with the use of interpolation, for the greatest number of cases.

Another approach to this problem is simply to generate the values of the table which are needed as the program progresses. Here the disadvantage is the complexity of the code which may be required to evaluate the function and the time required for the evaluation. The scientist may not be well versed in methods of function evaluation, particularly in terms of error estimation. Most programs which are written to evaluate functions are not transferrable between computers because of differences in machine languages.

One step toward the solution of some of these problems has been attempted here through the development of a group of FORTRAN programs which calculate various associated Legendre functions for arguments greater than 1. These programs are machine independent in the sense that FORTRAN is a language available on most computers and the modifications necessary to run on a different computer are relatively simple. The programs can also be considered as the first step toward a tabulation on tape since it would only be necessary to make the decisions mentioned above as to values of the argument and format of the tape and then obtain the function values from the programs described here.

Another problem concerned with tables of functions is the number of decimals to which the function should be calculated. Although a scientist usually does not require large numbers of decimals in specific

numerical problems, there are many areas where they are needed in order to have any significance in the final answer. This subject has been discussed at length in the Query-Reply section of Mathematical Tables and Other Aids to Computation.[1] It is particularly true in the case of associated Legendre functions for arguments greater than 1 where the function value is increasing or decreasing at a rapid rate compared with an increase in the value of the argument. Therefore it was considered desirable to allow for arbitrary precision in developing the programs since the type of use to which the function values will be put is the determining factor in the precision needed.

FOOTNOTES

1. "Q1, QR1: Tables to Many Places of Decimals," MTAC 1 (1943), pp. 30-31.

"QR2: Tables to Many Places of Decimals," MTAC 1 (1943), pp. 67-68.

"QR3: Tables to Many Places of Decimals," MTAC 1 (1943), p. 99.

"QR4: Tables to Many Places of Decimals," MTAC 1 (1943), pp. 99-100.

"QR27: Tables to Many Places of Decimals," MTAC 2 (1947), pp. 226-228.

CALCULATION OF ASSOCIATED LEGENDRE FUNCTIONS

The associated Legendre functions of the first and second kind, $P_\alpha^\mu(z)$ and $Q_\alpha^\mu(z)$, are solutions of the differential equation

$$(1-z^2)w''-2zw' + [\alpha(\alpha+1) - \frac{\mu^2}{1-z^2}]\, w=0 \ .$$

These solutions are usually represented in terms of the hypergeometric function.[1] We shall consider here only the cases $P_\alpha^m(x)$ and $Q_\alpha^m(x)$ where m is a non-negative integer, x is real and greater than 1 and $\alpha \neq -1, -2, -3, \ldots$ The Legendre functions can then be represented by the following definite integrals.[2]

$$P_\alpha^m(x) = \frac{\Gamma(\alpha+m+1)}{\pi\Gamma(\alpha+1)} \int_0^\pi (x + \sqrt{x^2-1}\cos t)^\alpha \cos mt\, dt$$

$$Q_\alpha^m(x) = (-1)^m \frac{\Gamma(\alpha+1)}{\Gamma(\alpha-m+1)} \int_0^\infty \frac{\cosh mt}{(x + \sqrt{x^2-1}\cosh t)^{\alpha+1}}\, dt$$

These integrals arise in many areas of applied mathematics, among them aerodynamics and radiation field studies.[3]

Both $P_\alpha^m(x)$ and $Q_\alpha^m(x)$ satisfy identical three-term recurrence relations of varying order m and degree α.[4] Of particular interest will be

$$U_{\alpha}^{m+1}(x) + \frac{2mx}{\sqrt{x^2-1}} U_{\alpha}^{m}(x) + (m+\alpha)(m-\alpha-1)U_{\alpha}^{m-1}(x)=0 \qquad 2.1$$

$$(\alpha-m+1)\, U_{\alpha+1}^{m}(x)-(2\alpha+1)xU_{\alpha}^{m}(x)+(\alpha+m)U_{\alpha-1}^{m}(x)=0 \qquad 2.2$$

where U stands for either P or Q.

The Legendre functions for $x > 1$ can be generated in a straight forward manner from these recurrence relations, they have in fact been used for previous table computation,[5] since the recursion is stable in the forward direction for $P_{\alpha}^{m}(x)$ and in the backward direction for $Q_{\alpha}^{m}(x)$. However, this method requires that starting values are available and these are difficult to obtain in some cases, particularly when α is not an integer or half-integer. The procedure to be described here, which replaces the condition of knowing two values of the function to begin with by a much more general condition in which at most one value must be known, was developed by Gautschi[6] based on the relationship between three-term recurrence relations and continued fractions. The algorithm is mathematically equivalent to the backward recurrence algorithm of J. C. P. Miller which has been used to obtain Legendre functions[7] although not as extensively or efficiently as by Gautschi.

A continued fraction can be related to every three-term recurrence relation in the following manner. Let A_n and B_n be solutions of

$$y_{n+1} + a_{n+1}y_n + b_{n+1}y_{n-1} = 0 \quad (n=0,1,2,\ldots) \qquad 2.3$$

determined by the initial values

$$A_{-1} = 1,\ A_0 = 0;\ B_{-1} = 0\quad B_0 = 1\ .$$

The A_n and B_n are then the numerators and denominators of the continued fraction

$$\frac{-b_1}{-a_1-}\ \frac{b_2}{-a_2-}\ \frac{b_3}{-a_3-}\ \cdots$$

or equivalently,

$$\frac{b_1}{a_1-}\ \frac{b_2}{a_2-}\ \frac{b_3}{a_3-}\ \cdots\ . \qquad 2.4$$

Alternatively, let

$$r_n = \frac{y_{n+1}}{y_n}\quad (n=-1,0,1,2,\ldots)\ .$$

The recurrence relation (2.3) can then be written as

$$r_n + a_{n+1} + \frac{b_{n+1}}{r_{n-1}} = 0,$$

or
$$r_{n-1} = \frac{-b_{n+1}}{a_{n+1} + r_n}$$

Repeating this for increasing n gives us

$$r_{n-1} = \frac{y_n}{y_{n-1}} = \frac{-b_{n+1}}{a_{n+1}-}\ \frac{b_{n+2}}{a_{n+2}-}\ \frac{b_{n+3}}{a_{n+3}-}\ \cdots\ .$$

In order to apply this relationship we need information on the convergence of the continued fraction and for what particular solution the ratios are to be formed. For this purpose we introduce the concept of distinguished solutions. A distinguished solution, f_n, is a solution of a difference equation (2.3) with the property that

$$\lim_{n\to\infty} \frac{f_n}{y_n} = 0$$

for any other linearly independent solution y_n of the difference equation. The following theorem[8] based on a result of Pincherle is then available.

Theorem 1 The continued fraction (2.4) converges if and only if the recurrence relation (2.3) possesses a distinguished solution f_n, with $f_{-1} \neq 0$. In case of convergence, moreover, one has

$$\frac{f_n}{f_{n-1}} = \frac{-b_{n+1}}{a_{n+1}-} \frac{b_{n+2}}{a_{n+2}-} \frac{b_{n+3}}{a_{n+3}-} \cdots \quad (n=0,1,2,\ldots),$$

provided $f_n \neq 0$ for $n = -1,0,1,2\ldots$.

Now write (2.3) in the form

$$y_{n+1} + a_n y_n + b_n y_{n-1} = 0 \quad (n = 1,2,3,\ldots) \qquad 2.5$$

and assume a nonvanishing distinguished solution, f_n, specified uniquely by one condition

$$\sum_{m=0}^{\infty} \lambda_m f_m = s \quad (\lambda_0 \neq 0,\ s \neq 0), \tag{2.6}$$

where s, and $\lambda_0, \lambda_1, \ldots,$ are given and the series converges and is normalized so that $\lambda_0 = 1$. If $\lambda_m = 0$ for all $m > 0$ this is the special case $f_0 = s$. The algorithm for computing f_n for $n = 0, 1, 2, \ldots, N$, based on Theorem 1 is then given as follows:[9]

Step 1: Select an integer $\nu > N$, and let $\phi_n^{(\nu)} = 0$ $(n = 0,1,\ldots,N)$.

Step 2: Calculate $f_n^{(\nu)}$ $(n = 0,1,\ldots,N)$ according to:

$$\left.\begin{aligned} r_{\nu}^{(\nu)} &= 0\ ,\quad r_{n-1}^{(\nu)} = \frac{-b_n}{a_n + r_n^{(\nu)}} \\ s_{\nu}^{(\nu)} &= 0\ ,\quad s_{n-1}^{(\nu)} = r_{n-1}^{(\nu)}\ (\lambda_n + s_n^{(\nu)}) \end{aligned}\right\} n=\nu,\nu-1,\ldots,1,$$

$$f_0^{(\nu)} = \frac{s}{1+s_0^{(\nu)}}\ ,\quad f_{n+1}^{(\nu)} = r_n^{(\nu)}\ f_n^{(\nu)}\ (n=0,1,\ldots,N-1). \tag{2.7}$$

Step 3: If the N+1 values of $f_n^{(\nu)}$ obtained in Step 2 do not agree with the current values of $\phi_n^{(\nu)}$ to within the desired accuracy, then redefine $\phi_n^{(\nu)}$ by $\phi_n^{(\nu)} = f_n^{(\nu)}$ $(n = 0,1,\ldots,N)$, increase ν by some integer, say 5, and repeat Step 2; otherwise accept $f_n^{(\nu)}$ as the final approximations to $f_n (n = 0,1,\ldots,N)$.

The convergence of this algorithm is stated by Gautschi in the following theorem.[10]

Theorem 2 Suppose the recurrence relation (2.5) has a nonvanishing distinguished solution, f_n, for which (2.6) holds. Let g_n be any other solution of (2.5). Then the algorithm (2.7) converges in the sense

$$\lim_{\nu \to \infty} f_n^{(\nu)} = f_n$$

if and only if

$$\lim_{\nu \to \infty} \frac{f_{\nu+1}}{g_{\nu+1}} \sum_{m=0}^{\nu} \lambda_m g_m = 0$$

is satisfied.

One remaining tool for applying this algorithm, coming from the asymptotic theory of difference equations, will allow us to show that a given solution of a three-term recurrence relation is a distinguished solution. The asymptotic structure of the general solution of a difference equation (2.5) whose coefficients satisfy

$$a_n \sim a n^{\alpha} , \quad b_n \sim b n^{\beta} \quad (ab \neq 0;\ \alpha, \beta \text{ real};\ n \to \infty)$$

depends on the Newton-Puiseux diagram formed with the points $P_0(0,0)$, $P_1(1,\alpha)$, $P_2(2,\beta)$. This is $\overline{P_0P_1P_2}$ if P_1 is above or on $\overline{P_0P_2}$, or $\overline{P_0P_2}$ otherwise. σ is the slope of $\overline{P_0P_1}$ ($\sigma=\alpha$) and τ is the slope of $\overline{P_1P_2}$ ($\tau=\beta-\alpha$). Gautschi now makes use of a theorem[11] from work by Perron and Kreuser.

Theorem 3 (a) If the point P_1 is above the line segment $\overline{P_0P_2}$ (i.e., $\sigma>\tau$), the difference equation (2.5) has two linearly independent solutions, $y_{n,1}$ and $y_{n,2}$, for which

$$\frac{y_{n+1,1}}{y_{n,1}} \sim -an^{\sigma}, \quad \frac{y_{n+1,2}}{y_{n,2}} \sim -\frac{b}{a}\, n^{\tau} \quad (n\to\infty) .$$

(b) If the points P_0, P_1, P_2 are collinear (i.e., $\sigma=\tau=\alpha$), let t_1,t_2 be the roots of $t^2+at+b=0$, and $|t_1| \geq |t_2|$. Then (2.5) has two linearly independent solutions, $y_{n,1}$ and $y_{n,2}$, such that

$$\frac{y_{n+1,1}}{y_{n,1}} \sim t_1 n^{\alpha}, \quad \frac{y_{n+1,2}}{y_{n,2}} \sim t_2 n^{\alpha} \ (n\to\infty) ,$$

provided $|t_1| > |t_2|$. If $|t_1| = |t_2|$ (in particular, if t_1,t_2 are complex conjugates, then

$$\limsup_{n\to\infty} \left[\frac{|y_n|}{(n!)^{\alpha}}\right]^{1/n} = |t_1|$$

for all solutions of (2.5).

(c) If the point P_1 lies below the line segment $\overline{P_0P_2}$ then

$$\limsup_{n\to\infty} \left[\frac{|y_n|}{(n!)^{\beta/2}}\right]^{1/n} = \sqrt{|b|}$$

for all solutions of (2.5)

It can be shown[12] that in case (a), and case (b) where $|t_1| > |t_2|$, the solution $f_n = y_{n,2}$ is a distinguished solution of (2.5). Also case (b) with $|t_1| > |t_2|$ gives us

$$\lim_{n\to\infty} \frac{y_{n+1}}{n^{\alpha} y_n} = t_r \quad (r = 1, \text{ or } r = 2),$$

where r = 2 for the distinguished solution, and r = 1 for any other solution.

To illustrate how this algorithm may be applied to the calculation of Legendre functions, consider the case of recurrence with respect to the order m where the degree α is not an integer. From (2.1) we see that $P_{\alpha}^{m}(x)$ and $Q_{\alpha}^{m}(x)$, as functions of m, are solutions of

$$y_{m+1} + \frac{2mx}{\sqrt{x^2-1}}\, y_m + (m+\alpha)(m-\alpha-1)y_{m-1} = 0 \qquad (m=1,2,3,\ldots)\,. \tag{2.8}$$

The Newton-Puiseux diagram of (2.8) is a straight line segment with slope 1. The roots of the equation

$$t^2 + \frac{2x}{\sqrt{x^2-1}}\, t + 1 = 0$$

are $$t_1 = -\sqrt{\frac{x+1}{x-1}}\,, \quad t_2 = t_1^{-1}$$

and since $x > 1$, $|t_1| > |t_2|$. Thus by case (b) of Theorem 3 we know that (2.8) possesses a distinguished solution f_m with

$$\lim_{m\to\infty} \frac{f_{m+1}}{m\, f_m} = t_2 .$$

Let $$F_m = \frac{P_\alpha^m(x)}{\Gamma(\alpha+m+1)} = \frac{1}{\pi\Gamma(\alpha+1)} \int_0^\pi [x+\sqrt{x^2-1}\cos t]^\alpha \cos mt\, dt.$$

Then $$\frac{F_{m+1}}{F_m} \sim \frac{P_\alpha^{m+1}(x)}{mP_\alpha^m(x)} \qquad (m\to\infty)$$

which has a finite limit, t_1 or t_2, as $m\to\infty$. If it were t_1, then $|F_m|$ would tend to ∞, since $|t_1| > 1$, but this is impossible by the definition of F_m. Therefore

$$\lim_{m\to\infty} \frac{P_\alpha^{m+1}(x)}{mP_\alpha^m(x)} = t_2$$

so $P_\alpha^m(x)$ is the distinguished solution of (2.8) and can be calculated for m = 0,1,2,... by the algorithm described, with the condition (2.6) given by the following identity,[13]

$$P_\alpha(x) + 2 \sum_{m=1}^{\infty} \frac{\Gamma(\alpha+1)}{\Gamma(\alpha+m+1)} P_\alpha^m(x) = (x + \sqrt{x^2-1}\,)^\alpha . \qquad 2.9$$

In the actual program, the algorithm is applied to the recurrence

$$F_{m+1} + \frac{2mx}{(m+\alpha+1)\sqrt{x^2-1}} F_m + \frac{m-\alpha-1}{m+\alpha+1} F_{m-1} = 0$$

which has $F_m = \frac{P_\alpha^m(x)}{\Gamma(\alpha+m+1)}$ as a distinguished solution. This simplifies the identity (2.9) to

$$F_0 + 2 \sum_{m=1}^{\infty} F_m = \frac{(x+\sqrt{x^2-1})^\alpha}{\Gamma(\alpha+1)}$$

We then have $\lambda_0 = 1$ and $\lambda_m = 2$ $(m = 1,2,\ldots)$ in our algorithm and can apply Theorem 2 to show convergence.

The application of the algorithm to other cases of the Legendre functions is similar and will be indicated in the next chapter as part of the description of the FORTRAN programs.

It should be noted that this theory is valid for $x = z$, a complex number outside the interval $(0,1)$ with $\mathrm{Re}\ z > 0$, although the programs here restrict x to a real number. Also of interest is the fact that the algorithm as applied above to the Legendre functions $P_\alpha^m(x)$ does not require the knowledge of any starting values which simplifies computation.

FOOTNOTES

1. M. Abramowitz and I. A. Stegun, ed., Handbook of Mathematical Functions, (Dover, New York, 1965), p. 332.

   E. W. Hobson, The Theory of Spherical and Ellipsoidal Harmonics, (University Press, Cambridge, 1931), pp. 89-118, 178-292.

2. H. Bateman, Higher Transcendental Functions, Vol. 1, (McGraw-Hill, New York, 1953), p. 157.

3. W. Gautschi, "Algorithm 229 Legendre Functions for Arguments Larger than One," Comm. Assoc. Comp. Mach. 8 (1965), p. 491.

4. Bateman, pp. 160-161.

   Abramowitz and Stegun, pp. 333-334.

5. NBS Mathematical Tables Project, Tables of Associated Legendre Functions, (Columbia University Press, New York, 1945).

6. W. Gautschi, Strength and Weakness of Three-Term Recurrence Relations, (Unpublished).

7. A Rotenberg, "The Calculation of Toroidal Harmonics," Math. Comput. 14 (1960), pp. 274-276.

8. Gautschi, Strength and Weakness, p. 11.

9. Ibid., p. 21.

10. Ibid., p. 24.

11. Ibid., p. 17.

12. Ibid., pp. 15-16, 18.

13. Bateman, p. 166.

III. MULTIPLE PRECISION PROGRAMS

In order to perform computations in multiple precision in a digital computer, it is necessary to define a system for storage of the multiple precision numbers and to have available subroutines to perform ordinary arithmetic operations such as add, multiply, etc. and also to evaluate common functions such as cosine or natural logarithm, since the regular operations and functions are, by implication of the name multiple precision, less accurate than needed. The basic arithmetic programs described here are a translation into FORTRAN by Henry C. Thacher, Jr. of Argonne National Laboratory of an ALGOL algorithm.[1] Their great advantage is that, being written in FORTRAN, they are easily transferred from one machine to another while most packages for multiple precision arithmetic are written in the assembly language of a specific computer and thus require a considerable expenditure of effort to use elsewhere. This advantage is somewhat offset by a large increase in the computer time necessary to run a specific program.

Each multiple precision number is stored in an integer array of arbitrary length K. This arbitrary array length is what allows the user to specify the precision desired. The number is stored in the form $a*(10^N)^b$ where N is chosen so that 10^{2*N} does not cause integer overflow. In our specific applications on an IBM 360 the maximum integer is 2147483647, and so N is set equal to 4. The number is standardized so that $1 \leq a < 10^N$. The multiple precision number $X(=a*(10^N)^b)$ is then stored as follows:

INTEGER*4 X(K)

X(1) = sign of the number: +1 if positive
-1 if negative
0 if zero (In this case, X(2) to X(K) are disregarded)

X(2) = exponent b of 10^N

X(3) = integral part of the mantissa, $1 \leq X(3) < 10^N$

X(4) to X(K) = fractional part of the mantissa, stored N decimal digits per location

Numbers stored in this manner have at least N*(K-3)+1 significant decimal digits. The maximum magnitude of a number which we can store on the 360 with N = 4 is $9999.99 \ldots *(10^4)^{2147483647} = 9.99 \ldots *10^{8589934591}$. This greatly extends the range of ordinary real numbers where the maximum magnitude is approximately 10^{75}.

The arithmetic operations are performed in floating point decimal arithmetic by a group of 13 subroutines. Listings and detailed descriptions of use are included in the Appendix. Since each subroutine which performs operations on the multiple precision numbers needs internal arrays for the temporary storage of numbers, it is found convenient to set a maximum value on K, rather than to pass each temporary array as a parameter. This maximum value was chosen as 50, which limits the number of decimal digits to 4*(50-3)+1=189, which was considered adequate for our purposes. This maximum K could be increased in the basic arithmetic subroutines by merely changing the DIMENSION statements, but would involve increasing the precision of certain constants in the function subroutines such as cosine, etc.

LNGCNS - This is the first subroutine of the multiple precision package. It initializes certain constants which are used by many of the other subroutines and stores them in labelled COMMON. Constants which must be supplied are N (=4 in our case) as described above, W(=8) the number of significant decimal digits in the representation of a real number, rounded up to the next higher integer if it is not an exact integer, and E(=75) the maximum allowable decimal exponent of a real number, rounded down to the next smaller integer if it is not an exact integer. This subroutine must be called before any of the basic arithmetic or function subroutines are used.

COPY - The parameters are A, B, and K where A and B are multiple precision numbers of array length K. This subroutine performs the operation B=A.

SET - This subroutine performs the operation A=I, where A is a multiple precision number and I is an ordinary integer. An error return occurs if the array A is not long enough to hold the integer I.

LNGTHN - This subroutine performs the operation A=X, where A is a multiple precision number and X is real (single precision). The accuracy of A is limited to single precision, i.e. the number of significant digits in A is the same as X, the remainder of the array being filled with 0's.

SHORTN - A multiple precision number A is converted to a single precision number X. An error return occurs if overflow would be produced.

ADJUST - A subroutine called internally by ADD and SBTRCT. It is essentially equivalent to a right shift.

READJT - A subroutine called internally by ADD and MLTINT. It re-adjusts a multiple precision number into standard form.

ADD - This subroutine performs the operation C = A + B, where A, B, and C are multiple precision numbers.

SBTRCT - Given A and B, this subroutine performs C = A - B, where A, B, and C are multiple precision numbers.

MLTPLY - This subroutine performs the operation C = A * B where A, B, and C are multiple precision numbers.

MLTINT - Given A, a multiple precision number, and I, an ordinary integer less than 10^N, this subroutine performs the operation B = A * I. It should be used instead of MLTPLY when possible since it is much faster (the time to execute MLTINT is proportional to K, the array length of a multiple precision number, while the execution time of MLTPLY is proportional to K^2). An error return occurs if $I \geq 10^N$.

RECIP - This subroutine finds the multiple precision reciprocal, B, of a multiple precision number, A, by an iterative procedure based on Newton's method for finding the root of a function, in this case 1/B - A = 0. B_0 is obtained by an ordinary division, 1.0/AS, where AS is the single precision equivalent of A obtained from the subroutine SHORTN. Therefore the regular system failure for division by zero will occur if A = 0. The exponent is handled separately so that underflow or overflow will not occur when A is within the limits of a multiple precision number, but AS would not be within the limits of an ordinary real number. The iteration proceeds by the formula

$B_{n+1} = B_n * (2 - B_n*A)$ and B_{n+1} is accepted as the final approximation if $B_{n+1} = B_n$ or $B_{n+1} = B_{n-1}$.

DIVIDE - Given the multiple precision numbers A and B, this subroutine will perform the operation C = A/B by using the subroutines RECIP and MLTPLY. For programming considerations, note that DIVIDE, by using RECIP, requires an iterative procedure while MLTPLY does not, so the programmer should avoid divisions whenever possible. For example, rather than dividing a multiple precision number by 2, multiply it by .5.

Besides these basic arithmetic operations, it is also necessary to be able to evaluate certain common functions. For the applications here it was necessary to evaluate $\sqrt{A}$, ln A, A^I, A^B, e^A, cos A, and $\Gamma(A)$, where A and B are multiple precision numbers and I is an ordinary integer. The first two programs were written by Thacher and modified only slightly by this author. The remaining are original programs.

LSQRT - This subroutine finds $Y = \sqrt{X}$ where X and Y are multiple precision numbers, by Newton's method for finding roots applied to the function $1 - X/Y^2 = 0$. This gives an iteration of $Y_{i+1} = Y_i + \delta_i$, where $\delta_i = Y_i(X - Y_i^2)/2X$, which minimizes divisions. Y_0 is obtained from the single precision square root routine and Y_{i+1} is accepted as the final approximation if $\delta_i = 0$ or $\exp(Y_{i+1}) - \exp(\delta_i) > K - 3$ (exp is the exponent, b, of 10^N in the multiple precision representation of a number stored in an array of length K). If $\exp(Y_{i+1}) - \exp(\delta_i) = K - 3$ and $\delta_i = -\delta_{i-1}$ then $Y_{i+1} = Y_i + \delta_i/2$ is accepted as the final

approximation to Y. If X < 0, the normal system error will occur in the single precision square root routine. An error message is printed if none of the conditions described are met for i $\leq$ 50.

LNGLOG - Y = ln X is evaluated by LNGLOG for X and Y multiple precision numbers. An error return occurs if X is nonpositive. The series

$$\ln X = 2^* \sum_{k=0}^{\nu-1} \left[\frac{1}{2k+1} \left(\frac{X-1}{X+1} \right)^{2k+1} \right] + \text{Rem} ,$$

$$\text{Rem} = \frac{2}{2\nu+1} \left(\frac{X-1}{X+1} \right)^{2\nu+1} / \left[1 - \left(\frac{X-1}{X+1} \right)^2 \right]$$

is used for .3 $\leq$ X $\leq$ 3. For other values of X, the range is reduced by using two constants, a tabulated value[2] of ln 10, and 4 * ln 10. The series is summed as follows:

$$S_0 = 2\left(\frac{X-1}{X+1} \right)$$

$$S_i = S_{i-1} + \delta_i \quad \text{where} \quad \delta_i = \frac{2}{2i+1} \left(\frac{X-1}{X+1} \right)^{2i+1}$$

until $\delta_i = 0$ or $\exp(S_{i-1}) - \exp(\delta_i) \geq K - 2$. If this criterion is not met for i $\leq$ 1500 an error message is printed.

OUTPUT - This is a subroutine used internally by LSQRT and LNGLOG to print error messages.

EXPO - This subroutine evaluates $Y = X^I$ where X and Y are multiple precision numbers and I is an ordinary integer. Y is obtained by multiplying X by itself $|I|$ times and then taking the reciprocal of the result if $I < 0$. An error message is printed if $X = I = 0$.

LEXPO - This subroutine evaluates $Y = X^A$ where X, Y, and A are multiple precision numbers. If A is an integer and less than or equal to $20*10^8$, the subroutine EXPO is used as it requires less execution time. Subroutines EXP and LNGLOG are used to find Y from the equation $Y = e^{A*\ln X}$. An error message is printed if $X = A = 0$ or if $X < 0$ and A is not an integer.

LEXP - Given the multiple precision number X, this subroutine will find $Y = e^X$ for $X \leq 20*10^8$. For $0 < X < .5$, e^X is evaluated from the continued fraction[3]

$$e^X = \frac{a_1}{b_1+}\ \frac{a_2}{b_2+}\ \frac{a_3}{b_3+}\ \cdots = \frac{1}{1-}\ \frac{X}{1+}\ \frac{X}{2-}\ \frac{X}{3+}\ \frac{X}{2-}\ \frac{X}{5+}\ \frac{X}{2-}\ \cdots$$

by the following algorithm[4]

$$u_1 = 1 \qquad v_1 = w_1 = \frac{a_1}{b_1}$$

$$u_{i+1} = \frac{1}{1 + \frac{a_{i+1}}{b_i b_{i+1}}\ u_i}\ , \quad v_{i+1} = v_i(u_{i+1}-1),\ w_{i+1}=w_i+v_{i+1} \qquad (i=1,2,3,\ldots)$$

If we write $e^X = \lim_{n\to\infty} \frac{A_n}{B_n}$ then $w_i = \frac{A_i}{B_i}$ and i is increased until $\exp(w_{i+1}) - \exp(v_{i+1}) \geq K - 2$. Then w_i is taken as the final approximation to e^X. If this criterion is not met for $i \leq 1000$ an error message is printed. An error message is also printed for $|X| > 20*10^8$. For other values of X, the range is reduced by using a tabulated value[5] of e^1.

LCOS - This subroutine finds Y = cos X for X and Y multiple precision numbers. X is first reduced to the interval $0 \leq X \leq 2\pi$ and then, using trigonometric reduction formulas, cos X is found by evaluating one of the following infinite series for $0 \leq X \leq \frac{\pi}{4}$.

$$\cos X = 1 - \frac{X^2}{2!} + \frac{X^4}{4!} - \frac{X^6}{6!} + \ldots$$

$$\sin X = X - \frac{X^3}{3!} + \frac{X^5}{5!} - \frac{X^7}{7!} + \ldots$$

The reductions are made using a constant value of 2π obtained from a tabulated value[6] of π. The series is summed until exp(Sum) - exp(term) $\geq$ K - 2. If this criterion is not met by the 1000th term an error message is printed. The series is summed ten terms at a time with the addition made from the smallest value to the largest in each group of ten to cut down on loss of significant figures.

LGAM - Y = $\Gamma(X)$, where X and Y are multiple precision numbers of array length K, is evaluated to D significant digits. D should be less than or equal to (K-3)*4+1 and X must be greater than 0. $\Gamma(X)$ is computed from the equations

$$\ln \Gamma(X+j) = (X+j-\tfrac{1}{2}) * \ln(X+j)-(X+j)+ \ln\sqrt{2\pi}+ \sum_{i=1}^{n} \frac{c_i}{(X+j)^{2i-1}}$$

3.1

where $c_i = \dfrac{(-1)^{i-1} B_i}{2i(2i-1)}$ and B_i is the ith Bernoulli number,

and $$\Gamma(X) = \frac{e^{\ln \Gamma(X+j)}}{\prod_{i=1}^{j-1} (x+i)} \quad .$$ 3.2

The values of j and n are chosen so as to give the desired D significant decimal digits.

The first equation above comes from the Sterling asymptotic expansion[7]

$$\ln \Gamma(t) = (t-\tfrac{1}{2})\ln t - t + \ln\sqrt{2\pi} + \phi(t)$$

where $$\phi(t) = \sum_{i=1}^{n} \frac{(-1)^{i-1}B_i}{2i(2i-1)} \frac{1}{t^{2i-1}} + R_n(t) ,$$

with $$R_n(t) < \frac{B_{n+1}}{2(n+1)(2n+1)} \cdot \frac{1}{t^{2n+1}} \quad \text{for } t > 0 .$$

It is seen from this that the error in $\Gamma(X)$ will depend on n and t. Since $R_n(t)$ decreases as t increases it is only necessary to determine the minimum value t^*, corresponding to a given n, for which $R_n(t)$ is small enough (i.e. $R_n(t) < 10^{-D}$) and then $R_n(t)$ will also be small

enough for all $t \geq t^*$. For a precision of up to 50 decimal digits, Filho and Schwachheim[8] determined, by a rough empirical relation, that computation time for $\ln \Gamma(t)$ could be minimized by using n=20 and $t^* = 7$ if $D < 17$ or $t^* = D-10$ otherwise. These values of n and t^* are used by LGAM for $D \leq 50$, but to extend D to 189, the maximum D since K is limited to 50 in these programs, it was desired to find the values of n so that the formula $t^* = D-10$ would still apply. Since we wanted to find n such that

$$\frac{B_{n+1}}{2(n+1)(2n+1)} \cdot \frac{1}{(t^*)^{2n+1}} \leq 10^{-(t^*+10)} \qquad 3.3$$

graphs were made of

$$\log y = (2n+1)\log t - t - 10 - \log \left[\frac{B_{n+1}}{2(n+1)(2n+1)} \right]$$

for various n. For the values of t where $\log y \geq 0$ the inequality (3.3) is satisfied. TABLE 1 is a summary of the values of n and t* chosen for use in LGAM.

TABLE 1

D	n	t^*
1 to 17	20	7
18 to 50	20	D-10
51 to 80	31	D-10
81 to 110	41	D-10
111 to 137	51	D-10
138 to 165	61	D-10
166 to 189	71	D-10

Given D, LGAM selects the proper n from TABLE 1. Then if X is greater than or equal to the corresponding t^* given in TABLE 1, $Y = \Gamma(X)$ can be computed directly from (3.1). For X less than the corresponding t^*, equations (3.1) and (3.2) are used with j the least integer which satisfies $X + j \geq t^*$.

The constants needed in equation (3.1) are $\ln\sqrt{2\pi}$, which was calculated from tabulated values of $\ln 2$[9] and $\ln \pi$[10], and the c_i, which were calculated using a table[11] of the numerators and denominators of the Bernoulli numbers.

The function subroutines were not extensively tested individually since the accuracy of the programs which calculate the Legendre functions using these function subroutines reflects their accuracy. (The tests of the Legendre programs are described in the next chapter.) However it was considered informative to check the function subroutines using a few known constants.[12] Two tests were run for each subroutine. Table 2 shows the relative error of the computed answer. D is the number of significant decimal digits which can be stored in a multiple precision number of array length K.

The seven subroutines which calculate the Legendre functions are translations of ordinary precision ALGOL programs written by Gautschi[13] into multiple precision FORTRAN programs using the multiple precision subroutines described above. The logic of the programs remains exactly the same with one exception. The starting value of ν in algorithm (2.7) was determined empirically by Gautschi[14]

TABLE 2

SUBROUTINE	TEST	RELATIVE ERROR				
		K=10 D=29	K=20 D=69	K=30 D=109	K=40 D=149	K=50 D=189
LSQRT	$\sqrt{\pi^2} = \pi$	$.32*10^{-28}$	$.32*10^{-68}$	0	0	0
$\sqrt{X}$	$\sqrt{e^8} = e^4$	$.18*10^{-29}$	0	$.18*10^{-109}$	$.18*10^{-149}$	0
LNGLOG	$\ln(e^{-1}) = -1$	$.20*10^{-27}$	$.10*10^{-67}$	$.10*10^{-107}$	$.30*10^{-147}$	$.10*10^{-187}$
ln X	$\ln(e^{10}) = 10$	$.10*10^{-28}$	$.10*10^{-68}$	$.10*10^{-108}$	$.40*10^{-148}$	$.20*10^{-188}$
EXPO	$(e^{-1})^{-2} = e^2$	$.54*10^{-28}$	$.41*10^{-68}$	0	$.54*10^{-148}$	$.41*10^{-188}$
X^I	$(e)^{10} = e^{10}$	$.18*10^{-27}$	$.23*10^{-67}$	$.32*10^{-107}$	$.18*10^{-147}$	$.14*10^{-187}$
LEXPO	$(e^{-10})^{-.4} = e^4$	$.12*10^{-27}$	$.11*10^{-67}$	$.62*10^{-108}$	$.84*10^{-148}$	$.48*10^{-188}$
X^A	$(e^4)2.5 = e^{10}$	$.18*10^{-27}$	$.24*10^{-66}$	$.73*10^{-107}$	$.19*10^{-146}$	$.11*10^{-186}$
LEXP	$e^{\ln 2} = 2$	$.25*10^{-27}$	$.55*10^{-67}$	$.55*10^{-107}$	$.55*10^{-147}$	$.50*10^{-188}$
e^X	$e^{\ln 9901} = 9901$	$.35*10^{-27}$	$.22*10^{-67}$	$.46*10^{-107}$	$.48*10^{-147}$	$.64*10^{-187}$
LCOS	$\cos(-1) = \cos 1$	$.96*10^{-30}$	$.35x10^{-69}$	$.15*10^{-109}$	$.87*10^{-150}$	$.37*10^{-191}$
cos X	$\cos(10) = \cos 10$	$.45*10^{-28}$	$.26*10^{-69}$	$.48*10^{-108}$	$.22*10^{-148}$	$.20*10^{-187}$
LGAM	$\Gamma(101) = 100!$	$.28*10^{-25}$	$.31*10^{-65}$	$.25*10^{-105}$	$.49*10^{-145}$	$.80*10^{-184}$
$\Gamma(X)$	$\Gamma(3/2) = \frac{1}{2}\sqrt{\pi}$	$.43*10^{-27}$	$.11*10^{-65}$	$.73*10^{-107}$	$.53*10^{-145}$	$.48*10^{-184}$

based on fixed values of nmax = 50 and d, the number of significant digits desired, equal to 6 and then is incremented by a constant value. Since our applications in multiple precision allow a much larger d, ν was seriously underestimated in cases of large d and with a constant increment much time was wasted before ν was in the correct range. Therefore the programs were changed to have the increment doubled at each step thus driving the value of ν up at a faster rate.

ILEG1 - This subroutine generates the associated Legendre function of the first kind

$$P_m^n(X) = \frac{(X^2-1)^{n/2}}{2^m m!} \frac{d^{m+n}}{dX^{m+n}} (X^2-1)^m ,$$

for n = 0, 1, ..., nmax given $m \geq 0$ and $X \geq 1$, where m and n are integers and X is a real, multiple precision number. The resulting P's are multiple precision.

For the special cases X = 1 or m = 0, $P_m^0(X) = 1$ and $P_m^n(X) = 0$ for n=1,2,..., nmax.

For other cases define $F_n = \frac{P_m^n(X)}{(n+m)!}$. The F_n then satisfy the recurrence relation

$$F_{n+1} + \frac{2nX}{(n+m+1)\sqrt{X^2-1}} F_n + \frac{(n-m-1)}{(n+m+1)} F_{n-1} = 0 \quad (n=1,2,3,\ldots$$

and the identity $F_0 + 2 \sum_{n=1}^{m} F_n = \frac{(X+\sqrt{X^2-1})^m}{m!}$

Since $P_m^n(X) = 0$ if $n > m$, $F_n = 0$ for $n > m$ and so the recurrence relation is related to the finite continued fraction

$$(n+m)\frac{F_n}{F_{n-1}} = \frac{(n+m)(m+1-n)}{nX_1+} \quad \frac{(n+m+1)(m-n)}{(n+1)X_1+} \quad \frac{(n+m+2)(m-n-1)}{(n+2)X_1+} \cdots \frac{2m\cdot 1}{mX_1} \quad (1 \leq n \leq m)$$

where $X_1 = 2X\sqrt{X^2-1}$. The algorithm (2.7) can now be applied with $\nu = m$ and no iteration is needed since $r_m = \frac{F_{m+1}}{F_m} \equiv 0$. The program proceeds as follows:

$$\left.\begin{aligned} R_m &= 0,\ R_{n-1} = \frac{(m+1-n)}{nX_1+(n+m+1)R_n} \\ S_m &= 0,\ S_{n-1} = R_{n-1}(2 + S_n) \end{aligned}\right\} \quad n=m,m-1,\ldots,1$$

$$P_m^0(X) = m!F_0 = m!\left[\frac{\frac{(X+\sqrt{X^2-1})^m}{m!}}{1+S_0}\right]$$

$$P_m^{n+1} = (m+n+1)!\ F_{n+1} = (m+n+1)\ P_m^n\ R_n \quad (n=0,\ 1,\ \ldots,\ m-1).$$

If nmax < m the last line above is calculated only for n = 0,1,..., nmax-1. If nmax > m, P_m^n is set equal to 0 for n = m+1, m+2, ...,nmax.

Accuracy is affected if X is very close to 1 since the computation of X_1 is subject to cancellation of significant digits. An error return occurs if X < 1 or m < 0 or nmax < 0.

ILEG2 - This subroutine generates to d significant digits the associated Legendre functions of the second kind $Q_n^m(X)$ for n=0,1,...,nmax given $m \geq 0$ and $X > 1$, where m and n are integers and X is a real multiple precision number. The resulting Q's are multiple precision.

The program first generates $Q_0^m(X)$ from the recurrence relation

$$Q_n^{i+1}(X) + \frac{2iX}{\sqrt{X^2-1}}\, Q_n^i(X) + (i+n)(i-n-1)Q_n^{i-1}(X) = 0 \quad (i=1,2,\ldots,m-1)$$

with n = 0 and the starting values $Q_0^0(X) = \frac{1}{2} \ln \frac{X+1}{X-1}$ and $Q_0^1(X) = \frac{-1}{\sqrt{X^2-1}}$.

Now let $F_n = Q_n^m(X)$ and apply algorithm (2.7) to the recurrence relation

$$(n-m+1)\, F_{n+1} - (2n+1)X\, F_n + (n+m)\, F_{n-1} = 0 \quad (n = 1,2,3,\ldots)$$

with the identity $F_o + \sum_{i=1}^{\infty} 0 \cdot F_i = Q_o^m(X)$.

Step 1: $\nu = 20 +$ integer part of $(5/4 * nmax)$

$inc = 10$

$\phi_n^{(\nu)} = 0 \quad (n=0,1,\ldots nmax)$

Step 2: $R_\nu^{(\nu)} = 0$, $R_{n-1}^{(\nu)} = \dfrac{(n+m)}{(2n+1)\, X - (n-m+1)\, R_n^{(\nu)}} \quad (n=\nu,\nu-1,\ldots,1)$,

$F_0^{(\nu)} = Q_0^m(X)$, $F_{n+1}^{(\nu)} = R_n^{(\nu)}\, F_n^{(\nu)} \quad (n = 0, 1, \ldots, nmax-1)$.

Step 3: If $\frac{|F_n^{(\nu)} - \phi_n^{(\nu)}|}{|F_n^{(\nu)}|} > .5*10^{-d}$ for any n = 0, 1, ..., nmax

then $\phi_n^{(\nu)} = F_n^{(\nu)}$ (n=0,1,...,nmax),

$\nu = \nu + inc$,

inc = 2*inc,

go to Step 2.

Otherwise accept F_n as the final approximation to

$Q_n^m(X)$ for n = 0,1,...,nmax.

Convergence of this algorithm is slow for X near 1. An error return occurs if $X \leq 1$ or nmax < 0 or m < 0.

ILEG 3 - This subroutine generates to d significant digits the associated Legendre function of the second kind $Q_n^m(X)$ for m=0,1,..., mmax given $n \geq 0$ and $X > 1$ where m and n are integers and X is a real multiple precision number. The resulting Q's are multiple precision.

The program first calls ILEG2 twice with parameters X, m=0 and nmax = n and X, m = 1 and nmax = n to obtain $Q_n^0(X)$ and $Q_n^1(X)$. The remaining Q's are generated from the recurrence relation

$$Q_n^{m+1}(X) + \frac{2mX}{\sqrt{X^2-1}} Q_n^m(X) + (m+n)(m-n-1)Q_n^{m-1}(X) = 0 \quad (m=1,2,...,mmax-1)$$

using the initial values $Q_n^0(X)$ and $Q_n^1(X)$ obtained from ILEG2.

An error return occurs if $n < 0$ or $mmax < 0$ or $X \leq 1$.

LEG1 - This subroutine generates to d significant digits the associated Legendre functions of the first kind

$$P_A^n(X) = \frac{\Gamma(A+n+1)}{\pi\Gamma(A+1)} \int_0^{\pi} [X + \sqrt{X^2-1}\cos t]^A \cos nt \, dt$$

for $n = 0,1,\ldots,nmax$ given $X \geq 1$ and A where n is an integer and X and A are real multiple precision numbers and A is not an integer. (ILEG1 should be used when A is an integer). The resulting P's are multiple precision.

For the special case $X = 1$, $P_A^0(X) = 1$ and $P_A^n(X) = 0$ for $n = 1,2,\ldots,$ nmax.

If $A < -\frac{1}{2}$, A can be replaced by $-A-1$ since $P_A^n(X) = P_{-A-1}^n(X)$. This substitution is made to avoid loss of accuracy when X is large.

Let $F_n = \dfrac{P_A^n(X)}{\Gamma(A+n+1)}$ and then algorithm (2.7) is applied to the recurrence relation

$$F_{n+1} + \frac{n}{(n+A+1)} X_1 F_n + \frac{(n-A-1)}{(n+A+1)} F_{n-1} = 0 \quad (n=1,2,3,\ldots)$$

where $X_1 = \dfrac{2X}{\sqrt{X^2-1}}$

and the identity $F_0 + 2 \sum_{n=1}^{\infty} F_n = \dfrac{[X + \sqrt{X^2-1}]^A}{\Gamma(A+1)}$

Step 1: $\nu = 20 + \text{integer part of } \left\{\frac{[37.26+.1283(A+38.26)X]}{37.26 + .1283(A+1)X}\right\}*nmax$

$inc = 10$

$\phi_n^{(\nu)} = 0 \ (n=0,1,\ldots,nmax)$

Step 2:
$$\left.\begin{aligned} R_\nu^{(\nu)} &= 0, \ R_{n-1}^{(\nu)} = \frac{(A+1-n)}{nX_1+(n+A+1)R_n^{(\nu)}} \\ S_\nu^{(\nu)} &= 0, \ S_{n-1}^{(\nu)} = R_{n-1}^{(\nu)} \ (2 + S_n^{(\nu)}) \end{aligned}\right\} n = \nu, \nu-1, \ldots, 1,$$

$$F_0^{(\nu)} = \frac{\frac{(X+\sqrt{X^2-1})^A}{\Gamma(A+1)}}{1 + S_0^{(\nu)}}, \quad F_{n+1}^{(\nu)} = R_n^{(\nu)} F_n^{(\nu)} \ (n=0,1,\ldots,nmax-1).$$

Step 3: If $\frac{|F_n^{(\nu)} - \phi_n^{(\nu)}|}{|F_n^{(\nu)}|} > .5*10^{-d}$ for any $n = 0,1,\ldots,nmax$

then $\phi_n^{(\nu)} = F_n^{(\nu)} \ (n = 0,1,\ldots,nmax)$,

$\nu = \nu+inc$,

$inc = 2*inc$,

go to Step 2.

Otherwise $P_A^n(X) = \Gamma(A+n+1)*F_n \ (n=0,1,\ldots,nmax)$.

Accuracy is affected if X is very close to 1 since the computation of X_1 is subject to cancellation of significant digits. The rate of convergence of the algorithm decreases as X increases. An error return occurs if $X < 1$ or $nmax < 0$ or A is an integer.

LEG2 - This subroutine evaluates to d significant digits the associated Legendre functions of the first kind $P^m_{A+n}(X)$ for n = 0,1,...,nmax given $X \geq 1$, A, and $m \geq 0$ where m and n are integers and X and A are real multiple precision numbers and A is not an integer. The resulting P's are multiple precision.

The program first calls LEG1 twice with parameters X, A and m and X, A+1 and m to obtain $P_A^m(X)$ and $P^m_{A+1}(X)$. The remaining P's are generated from the recurrence relation

$$P^m_{A+n+1}(X) - \frac{2n+2A+1}{n+A-m+1} \; X \; P^m_{A+n}(X) + \frac{n+A+m}{n+A-m+1} \; P^m_{A+n-1}(X) = 0 \quad (n=0,1,\ldots, \text{nmax}-1)$$

using the initial values $P^m_A(X)$ and $P^m_{A+1}(X)$ obtained from LEG1.

An error return occurs if $m < 0$ or $X < 1$ or nmax < 0 or A is an integer.

CONIC - This subroutine generates to d significant digits Mehler's conical functions $P^n_{-1/2+iT}(X)$, a special case of the associated Legendre function of the first kind, for n = 0,1,...,nmax given $X \geq 1$ and T where n is an integer and X and T are real multiple precision numbers. The resulting P's are multiple precision numbers.

For the special case X = 1, $P^0_{-1/2+iT}(X) = 1$ and $P^n_{-1/2+iT}(X) = 0$ for n= 1,2,...,nmax.

For other X, let $F_n = \dfrac{P_{-1/2+iT}^{n}(X)}{n!}$ and apply algorithm (2.7)

to the recurrence relation

$$F_{n+1} + \frac{nX_1}{(n+1)} F_n + \frac{(n-\frac{1}{2})^2 + T^2}{n(n+1)} F_{n-1} = 0 \quad (n = 1,2,\ldots)$$

where

$$X_1 = \frac{2X}{\sqrt{X^2-1}}$$

and the identity $F_0 + \sum_{n=1}^{\infty} \lambda_n F_n = \dfrac{\cos[T \ln(X+\sqrt{X^2-1})]}{\sqrt{X+\sqrt{X^2-1}}}$

where $\lambda_n = n!\left[\dfrac{\Gamma(\frac{1}{2}+iT)}{\Gamma(\frac{1}{2}+iT+n)} + \dfrac{\Gamma(\frac{1}{2}-iT)}{\Gamma(\frac{1}{2}-iT+n)}\right]$

Step 1: ν = 30 + interger part of {[1+(.140+.0246T)(X-1)]nmax}

inc = 60

$\phi_n^{(\nu)} = 0$ (n=0,1,...,nmax).

Step 2: Obtain $\lambda_{\nu+1}^{(\nu)}$, $\lambda_{\nu}^{(\nu)}$ from

$$\lambda_1^{(\nu)} = \frac{1}{\frac{1}{4}+T^2}, \quad \lambda_2^{(\nu)} = \frac{3-4T^2}{(\frac{1}{4}+T^2)(\frac{9}{4}+T^2)},$$

$$\lambda_{n+1}^{(\nu)} = \frac{1+\frac{1}{N}}{(1+\frac{1}{2n})^2+(\frac{T}{n})^2}\,(2\lambda_n-\lambda_{n-1}) \quad (n=2,3,\ldots,\nu).$$

$$\left.\begin{aligned} R_{\nu}^{(\nu)} &= 0, \quad R_{n-1}^{(\nu)} = -\frac{(1-\frac{1}{2n})^2+(\frac{T}{n})^2}{X_1+(1+\frac{1}{n})R_n^{(\nu)}} \\ S_{\nu}^{(\nu)} &= 0\ ,\ S_{n-1}^{(\nu)} = R_{n-1}^{(\nu)}\,(\lambda_n + S_n^{(\nu)}) \\ \lambda_{n-1}^{(\nu)} &= 2\lambda_n - \frac{(1+\frac{1}{2n})^2+(\frac{T}{n})^2}{1+\frac{1}{n}}\,\lambda_{n+1} \end{aligned}\right\} n=\nu\ ,\ \nu-1,\ldots,1$$

$$F_0^{(\nu)} = \frac{\cos[T * \ln(X+\sqrt{X^2-1})]}{\sqrt{X+\sqrt{X^2-1}}}\ ,\ F_{n+1}^{(\nu)} = R_n^{(\nu)}\,F_n^{(\nu)} \quad (n=0,1,\ldots,\text{nmax}-1).$$

Step 3: If $\dfrac{|F_n^{(\nu)} - \phi_n^{(\nu)}|}{|F_n^{(\nu)}|} > .5*10^{-d}$ for any $n=0,1,\ldots,\text{nmax}$

then $\phi_n^{(\nu)} = F_n^{(\nu)}$ $(n=0,1,\ldots,\text{nmax})$,

$\nu = \nu + \text{inc}$,

inc = 2*inc,

go to Step 2.

Otherwise $P^n_{-1/2+i\mathbf{T}}(X) = n!F_n$ $(n=0,1,\ldots,\text{nmax})$.

This algorithm converges slowly when X and T are both large. It may not converge at all due to accumulation of roundoff errors if d is too large. Therefore the iteration is terminated and an error message is printed if ν becomes greater than 1500. An error return occurs if X < 1 or nmax < 0.

TOROID - This subroutine generates to d significant digits the toroidal functions of the second kind $Q^m_{-1/2+n}(X)$ for n=0,1,...,nmax given X > 1 a multiple precision number and m an integer. The resulting Q's are multiple precision.

Let $F_n = Q^m_{-1/2+n}(X)$. Algorithm (2.7) is then applied to the recurrence relation

$$(n-m+\tfrac{1}{2})\,F_{n+1} - 2nX\,F_n + (n+m-\tfrac{1}{2})\,F_{n-1} = 0 \quad (n=1,2,3,\ldots)$$

and the identity $F_0 + 2\sum_{n=1}^{\infty} F_n = (-1)^m \sqrt{\frac{\pi}{2}}\ \Gamma(m+\frac{1}{2})(X-1)^{-\frac{1}{2}} \left(\frac{X+1}{X-1}\right)^{\frac{m}{2}}$.

Step 1: ν = 20 + integer part of $\{[1.15+(.0146+.00122m)/(X-1)]nmax\}$

inc = 10

$\phi_n^{(\nu)} = 0$ (n=0,1,...,nmax).

Step 2: $R_\nu^{(\nu)} = 0$, $R_{n-1}^{(\nu)} = \dfrac{n+m-\frac{1}{2}}{2nX-(n-m+\frac{1}{2})R_n^{(\nu)}}$

$S_\nu^{(\nu)} = 0$, $S_{n-1}^{(\nu)} = R_{n-1}^{(\nu)}\ (2+S_n^{(\nu)})$

$\Big\}$ $n = \nu, \nu-1, \ldots, 1,$

$$F_0^{(\nu)} = \frac{(-1)^m \sqrt{\frac{\pi}{2}}\, \Gamma(m+\tfrac{1}{2})(X-1)^{-\frac{1}{2}} \left(\frac{X+1}{X-1}\right)^{\frac{m}{2}}}{1 + S_0^{(\nu)}}, \quad F_{n+1}^{(\nu)} = R_n^{(\nu)} F_n^{(\nu)}$$

$(n=0,1,\ldots,nmax$

Step 3: If $\dfrac{|F_n^{(\nu)} - \phi_n^{(\nu)}|}{|F_n^{(\nu)}|} > .5*10^{-d}$ for any n=0,1,...,nmax

then $\phi_n^{(\nu)} = F_n^{(\nu)}$ (n=0,1,...,nmax),

$\nu = \nu + inc$,

inc = 2*inc,

go to Step 2.

Otherwise accept F_n as the final approximation to

$Q^m_{-1/2+n}(X)$ for n=0,1,...,nmax.

Convergence is slow for X near 1. An error return occurs for

$X \leq 1$ or nmax < 0.

FOOTNOTES

1. I.D. Hill, "Algorithm 34 Procedures for the Basic Arithmetical Operations in Multiple-Length Working," Computer J. 11 (1968), pp. 232-235.

2. H. S. Uhler, Original Tables to 137 Decimal Places of Natural Logarithms for Factors of the Form $1 \pm n \cdot 10^{-p}$, Enhanced by Auxiliary Tables of Logarithms of Small Integers, (New Haven, Connecticut, 1942), Table 4.

3. Abramowitz and Stegun, p. 70.

4. Gautschi, Strength and Weakness, pp. 9-10.

5. "Review 275," MTAC 2 (1946), p. 69.

6. "A New Approximation to π," MTAC 2 (1947), p. 247.

7. H.Werner and R. Collinge, "Chebyshev Approximations to the Gamma Function," Math. Comput. 15 (1961) p. 195.

8. A. M. S. Filho and G. Schwachheim, "Algorithm 309 Gamma Function with Arbitrary Precision," Comm. Assoc. Comp. Mach. 10 (1967), pp. 511-512.

9. H. S. Uhler, "Recalculation and Extension of the Modulus and of the Logarithms of 2, 3, 5, 7 and 17," Proc. Nat. Acad. Sci. 26 (1940), p. 209.

10. H. S. Uhler, "Log π and other Basic Constants," Proc. Nat. Acad. Sci. 24 (1938), p. 27.

11. H. T. Davis, Tables of Higher Mathematical Functions Vol. 2, (Bloomington, 1935), pp. 230-232.

12. H. S. Uhler, "A New Table of Reciprocals of Factorials and Some Derived Numbers," Trans. Conn. Acad. Arts. Sci. 32 (1937), p. 433-434.

Uhler, Proc. Nat. Acad. Sci. 24, p. 29.

Uhler, Proc. Nat. Acad. Sci. 26.

H. S. Uhler, "The Coefficients of Sterling's Series for Log $\Gamma(x)$," Proc. Nat. Acad. Sci. 28 (1942), p. 61.

H. S. Uhler, "Natural Logarithms of Small Prime Numbers," Proc. Nat. Acad. Sci. 29 (1943), p. 322.

13. Gautschi, Comm. Assoc. Comp. Mach. 8, pp. 488-492.

14. Gautschi, Strength and Weakness, pp. 56-60.

IV. DESCRIPTION OF TESTS AND EXPERIMENTAL RESULTS

The accuracy of the programs which calculate the Legendre functions was determined by a combination of two types of tests: 1) independent calculation and 2) satisfaction of recurrence relation or internal consistency. First an array of values of the particular function $U_a^n(x)$, where n is taken as the row index, is generated for various values of x. The error in one or more elements of the matrix is checked for selected key values of a and n by an independent calculation using the explicit expression in x or infinite series representation. The remaining elements are then checked by using the recurrence relation

$$U_a^{n+1}(x) = \frac{1}{\sqrt{x^2-1}} [(a-n)xU_a^n(x)-(a+n)U_{a-1}^n(x)] \qquad 4.1$$

to relate the error in the remaining elements to the predetermined error in the key elements or by "internal consistency", comparing similar results obtained from two different Legendre function programs.

The combination of tests is needed since in many cases neither of the two tests mentioned above are suitable for use alone. A program may be tested by independently calculating every value. However an explicit expression is generally known only for a few special values of a and n and the infinite series may converge so slowly that it would require too much time to test every value. Checking by means of a recurrence relation alone is unsatisfactory because of the chance of "error growth". For example, suppose the absolute error

in $U_a^n(x)$ is $n\varepsilon$ where ε is a small number. Then the error in elements of the nth row will differ only slightly from the error in the elements of the (n+1)st row and the recurrence relation would be satisfied. However, although the error in $U_a^1(x)$ is small, for large values of n, the error has "grown" large. Internal consistency is also inadequate for use by itself since the two programs being compared could both contain the same error since they are based on similar methods.

ILEG1 - ILEG1 generates the associated Legendre function of the first kind $P_m^n(x)$, for n=0,1,...,nmax given $m \geq 0$ and $x \geq 1$, where m and n are integers and x is real. For this special case where m is a non-negative integer,

$$P_m^n(x) = \frac{(x^2-1)^{n/2}}{2^m m!} \frac{d^{m+n}}{dx^{m+n}} (x^2-1)^m.$$

Thus if $n > m$, $P_m^n(x) = 0$ since $\frac{d^{m+n}}{dx^{m+n}} (x^2-1)^m = 0$. Because ILEG1 takes advantage of this fact and sets $P_m^n(x)$ equal to 0 for $n > m$, we need only consider the upper right triangular matrix, which contains the non-zero values of $P_m^n(x)$, for our testing purposes. An 11x11 matrix was chosen for study ($0 \leq n \leq m \leq 10$).

$$\begin{bmatrix} P_0^0 & P_1^0 \cdots & P_{10}^0 \\ & P_1^1 \cdots & P_{10}^1 \\ \text{0} & \ddots & \vdots \\ & & P_{10}^{10} \end{bmatrix}$$

The absolute error of the diagonal elements was determined by comparison with values computed by means of the explicit expression:

$$P_n^n(x) = (x^2-1)^{n/2} \prod_{i=1}^{n} (2n-2i+1) .$$

Since the maximum length of the arrays used to hold the long precision numbers in ILEG1 is 50 locations, the $P_n^n(x)$ were calculated by a program which used arrays of length 52 to eliminate the accumulation of rounding error in the 50th location. Using arrays of length 52 assures that the error in $P_n^n(x)$ is at most 1 in the least significant digit of the number, this coming from truncation of the final number.

The error in the other elements of the matrix can be related to the error of the diagonal elements by means of (4.1). The following table shows this relationship, where ε_n is the absolute error of $P_n^n(x)$, determined from the value of $P_n^n(x)$ computed by ILEG1 and by the explicit expression as described above.

$$\begin{bmatrix} \varepsilon_0 & \frac{\sqrt{x^2-1}}{x}\varepsilon_1 & \frac{x^2-1}{2x^2}\varepsilon_2 & \cdots & \frac{(x^2-1)^5}{10!x^5}\varepsilon_{10} \\ & \varepsilon_1 & \frac{\sqrt{x^2-1}}{x}\varepsilon_2 & & \vdots \\ & & \varepsilon_2 & & \\ & \text{0} & & & \frac{x^2-1}{2x^2}\varepsilon_{10} \\ & & & & \frac{\sqrt{x^2-1}}{x}\varepsilon_{10} \\ & & & & \varepsilon_{10} \end{bmatrix}$$

This table was arrived at in the following manner. Consider for example that $\{P_9^7\}^* = P_9^7 + \varepsilon$ and that all other values are exact. By (4.1) we have:

$$\{P_9^8\}^* = \frac{1}{\sqrt{x^2-1}} \quad [2x\{P_9^7\}^* - 16\ P_8^7]$$

$$\{P_9^8\}^* = \frac{1}{\sqrt{x^2-1}} \quad [2x(P_9^7 + \varepsilon) - 16\ P_8^7]$$

$$\{P_9^8\}^* = P_9^8 + \frac{2x}{\sqrt{x^2-1}}\ \varepsilon$$

Similarly,

$$\{P_9^9\}^* = \frac{1}{\sqrt{x^2-1}} \quad [x\{P_9^8\}^* - 17\ P_8^8]$$

$$\{P_9^9\}^* = \frac{1}{\sqrt{x^2-1}} \quad [x(P_9^8 + \frac{2x}{\sqrt{x^2-1}}\ \varepsilon) - 17\ P_8^8]$$

$$\{P_9^9\}^* = P_9^9 + \frac{2x^2}{x^2-1}\ \varepsilon$$

$$\varepsilon = \frac{x^2-1}{2x^2} \quad [\{P_9^9\}^* - P_9^9]$$

Now we know the value of $\varepsilon_9 = [\{P_9^9\}^* - P_9^9]$ so our conclusion is that if (4.1) is satisfied by the values in the matrix, then the error in $P_9^7(x)$,

$$\varepsilon_9^7 = \frac{x^2-1}{2x^2} \varepsilon_9 .$$

The table can also be expressed by the formula:

$$\varepsilon_m^n = \frac{(x^2-1)^{(m-n)/2}}{(m-n)!x^{(m-n)}} \varepsilon_m .$$

Since $x > 1$, $(x^2-1)^{(m-n)/2} < x^{m-n}$ and the following inequality holds:

$$\varepsilon_m^n < \frac{1}{(m-n)!} \varepsilon_m$$

Then if $P_{mmin} = \min_{0 \leq n \leq m} (m-n)! \ |P_m^n|$

we can calculate the maximum relative error for each column, η_m, by

$$\eta_m = \frac{\varepsilon_m}{P_{mmin}}$$

Table 3 shows the results of tests run. K is the number of locations in the array used for each multiple precision number and D is the corresponding number of digits which can be stored in an array with K locations. Maximum relative residual refers to the check of recurrence relation (4.1). It is defined as:

TABLE 3

ILEG1

K=10 D=29

m	x=1.1 r_{max}	x=1.1 η_m	x=5.0 r_{max}	x=5.0 η_m	x=10.0 r_{max}	x=10.0 η_m	x=25.0 r_{max}	x=25.0 η_m
0		0		0		0		0
1	$.42*10^{-27}$	$.87*10^{-28}$	$.61*10^{-28}$	$.41*10^{-28}$	$.10*10^{-28}$	$.40*10^{-28}$	$.88*10^{-28}$	$.60*10^{-28}$
2	$.56*10^{-27}$	$.94*10^{-28}$	$.11*10^{-27}$	$.22*10^{-28}$	$.76*10^{-28}$	$.40*10^{-29}$	$.11*10^{-27}$	$.91*10^{-28}$
3	$.21*10^{-27}$	$.21*10^{-27}$	$.10*10^{-27}$	$.68*10^{-28}$	$.14*10^{-27}$	$.14*10^{-27}$	$.13*10^{-27}$	$.39*10^{-28}$
4	$.42*10^{-27}$	$.32*10^{-27}$	$.96*10^{-28}$	$.17*10^{-28}$	$.70*10^{-28}$	$.10*10^{-27}$	$.14*10^{-27}$	$.16*10^{-27}$
5	$.11*10^{-26}$	$.13*10^{-27}$	$.11*10^{-27}$	$.21*10^{-28}$	$.79*10^{-28}$	$.58*10^{-28}$	$.15*10^{-27}$	$.12*10^{-27}$
6	$.16*10^{-27}$	$.20*10^{-27}$	$.14*10^{-27}$	0	$.90*10^{-28}$	$.12*10^{-28}$	$.16*10^{-27}$	$.24*10^{-27}$
7	$.33*10^{-27}$	$.29*10^{-27}$	$.12*10^{-27}$	$.12*10^{-27}$	$.77*10^{-28}$	$.31*10^{-27}$	$.12*10^{-27}$	$.71*10^{-28}$
8	$.24*10^{-27}$	$.36*10^{-27}$	$.14*10^{-27}$	$.21*10^{-27}$	$.61*10^{-28}$	$.12*10^{-27}$	$.12*10^{-27}$	$.16*10^{-27}$
9	$.46*10^{-27}$	$.33*10^{-27}$	$.80*10^{-28}$	$.70*10^{-28}$	$.92*10^{-28}$	$.15*10^{-27}$	$.15*10^{-27}$	$.23*10^{-27}$
10	$.25*10^{-27}$	$.30*10^{-27}$	$.10*10^{-27}$	$.54*10^{-28}$	$.76*10^{-28}$	$.84*10^{-28}$	$.12*10^{-27}$	$.23*10^{-27}$

K=20 D=69

m	x=1.1 r_{max}	x=1.1 η_m	x=5.0 r_{max}	x=5.0 η_m	x=10.0 r_{max}	x=10.0 η_m	x=25.0 r_{max}	x=25.0 η_m
0		0		0		0		0
1	$.70*10^{-68}$	$.56*10^{-68}$	$.14*10^{-67}$	$.41*10^{-68}$	$.91*10^{-68}$	$.10*10^{-68}$	$.10*10^{-67}$	$.16*10^{-68}$
2	$.27*10^{-67}$	$.11*10^{-68}$	$.92*10^{-68}$	$.22*10^{-68}$	$.14*10^{-67}$	$.97*10^{-68}$	$.11*10^{-67}$	$.11*10^{-67}$
3	$.21*10^{-67}$	$.69*10^{-68}$	$.97*10^{-68}$	$.96*10^{-68}$	$.83*10^{-68}$	$.68*10^{-68}$	$.94*10^{-68}$	$.68*10^{-68}$

TABLE 3 (Continued)

m	r_{max}	η_m	r_{max}	η_m	r_{max}	η_m	r_{max}	η_m
4	$.30*10^{-67}$	$.65*10^{-68}$	$.13*10^{-67}$	$.22*10^{-67}$	$.12*10^{-67}$	$.18*10^{-67}$	$.93*10^{-68}$	$.83*10^{-68}$
5	$.56*10^{-67}$	$.23*10^{-67}$	$.13*10^{-67}$	$.11*10^{-67}$	$.12*10^{-67}$	$.14*10^{-67}$	$.14*10^{-67}$	$.25*10^{-67}$
6	$.30*10^{-67}$	$.26*10^{-67}$	$.14*10^{-67}$	$.14*10^{-67}$	$.91*10^{-68}$	$.71*10^{-68}$	$.12*10^{-67}$	$.20*10^{-67}$
7	$.34*10^{-67}$	$.24*10^{-67}$	$.11*10^{-67}$	$.15*10^{-67}$	$.15*10^{-67}$	$.15*10^{-67}$	$.89*10^{-68}$	$.48*10^{-68}$
8	$.69*10^{-67}$	$.14*10^{-67}$	$.14*10^{-67}$	$.24*10^{-67}$	$.14*10^{-67}$	$.30*10^{-67}$	$.14*10^{-67}$	$.26*10^{-67}$
9	$.85*10^{-67}$	$.36*10^{-67}$	$.10*10^{-67}$	$.11*10^{-67}$	$.12*10^{-67}$	$.15*10^{-67}$	$.15*10^{-67}$	$.31*10^{-67}$
10	$.58*10^{-67}$	$.22*10^{-67}$	$.14*10^{-67}$	$.43*10^{-67}$	$.13*10^{-67}$	$.30*10^{-67}$	$.16*10^{-67}$	$.26*10^{-67}$

K = 50 D = 189

x = 5.0

m	r_{max}	η_m
0		0
1	0	$.41*10^{-188}$
2	$.97*10^{-188}$	$.44*10^{-188}$
3	$.78*10^{-188}$	$.41*10^{-188}$
4	$.33*10^{-188}$	$.17*10^{-188}$
5	$.73*10^{-188}$	$.26*10^{-189}$
6	$.70*10^{-188}$	$.70*10^{-188}$
7	$.77*10^{-188}$	$.55*10^{-188}$
8	$.14*10^{-187}$	$.27*10^{-188}$
9	$.79*10^{-188}$	$.25*10^{-187}$
10	$.88*10^{-188}$	$.11*10^{-187}$

$$r_{max} = \max_{0\leq n\leq m-1} \frac{\left| P_m^{n+1} - \frac{1}{\sqrt{x^2-1}} [(m-n)xP_m^n-(m+n)P_{m-1}^n] \right|}{|P_m^{n+1}|}$$

for $1\leq m\leq 10$ and calculated using the $P_m^n(x)$ obtained from ILEG1.

The number of significant digits calculated was (K-4)*4+3 (or D-2) in all cases except K = 10 and x = 1.1. Here one of the recurrence relation checks shows an error in the 27th digit indicating there may be only (K-4)*4+2 significant digits in this case. However this is most likely due to loss of accuracy in the division $\frac{1}{\sqrt{x^2-1}}$ in the calculation of r_{max} and we would still be justified in saying there are D-2 significant digits on the basis of the calculated relative error.

ILEG2 - ILEG2 generates to d significant digits the associated Legendre function of the second kind $Q_n^m(x)$, for n=0,1,...,nmax given $m \geq 0$ and $x > 1$, where m and n are integers and x is real. It was mentioned in the beginning of this chapter that a program could be tested by independently calculating every value but that generally an explicit expression was known only for a very few values of m and n. In the case of $Q_n^m(x)$, the explicit expressions have been generated for $0\leq n,m\leq 4$ by Suschowk.[1] These formulae were used, with the exception of Q_4^4 which is incorrect and should read

$$Q_4^4(x) = \frac{1}{(x^2-1)^2} [(105x^8 - 420x^6 + 630x^4-420x^2 + 105)*\ln[(x+1)/(x-1)]/2 - 105x^7 + 385x^5 - 511x^3 + 279x]$$

by a program which used arrays of length K+2 for storage of the multiple precision numbers so that accumulation of rounding error would not affect the accuracy of the numbers calculated. The results, $\{Q_n^m\}^*$, truncated to multiple precision numbers of array length K can then be considered as exact values and the relative error of the Q_n^m calculated by ILEG2 can be determined by

$$\eta_n^m = \left| \frac{\{Q_n^m\}^* - Q_n^m}{\{Q_n^m\}^*} \right| .$$

Table 4 gives the maximum relative error for each column generated by ILEG2,

$$\eta_m = \max_{0 \leq n \leq 4} \eta_n^m$$

for various values of x and K. D is the number of significant digits asked for by the calling program. The largest relative error observed was $.47*10^{-27}$ when x = 10.0, K = 10 and D = 28 indicating that when K is so close to the total number of digits which can be stored in the multiple precision number of array length K ((10-3)*4+1 = 29), there may only be D-1 significant digits in the result.

ILEG3 - ILEG3 generates to d significant digits the associated Legendre function of the second kind $Q_n^m(x)$, for m=0,1,...,mmax given $n \geq 0$ and $x > 1$, where m and n are integers and x is real.

TABLE 4

ILEG2

K=10 D=28

m	x=1.1 η_m	x=5.0 η_m	x=10.0 η_m	x=25.0 η_m
0	$.13*10^{-27}$	$.67*10^{-28}$	$.47*10^{-27}$	$.43*10^{-27}$
1	0	$.78*10^{-28}$	$.40*10^{-28}$	$.15*10^{-27}$
2	$.15*10^{-27}$	$.11*10^{-27}$	$.86*10^{-28}$	$.10*10^{-27}$
3	$.88*10^{-28}$	$.77*10^{-28}$	$.10*10^{-27}$	$.19*10^{-27}$
4	$.17*10^{-27}$	$.92*10^{-28}$	$.15*10^{-27}$	$.41*10^{-27}$

K=15 D=48

m	x=1.1 η_m	x=5.0 η_m	x=10.0 η_m	x=25.0 η_m
0	$.66*10^{-48}$	$.46*10^{-48}$	$.12*10^{-47}$	$.43*10^{-47}$
1	$.94*10^{-48}$	$.47*10^{-48}$	$.28*10^{-48}$	$.77*10^{-48}$
2	$.96*10^{-48}$	$.49*10^{-48}$	$.49*10^{-48}$	$.64*10^{-48}$
3	$.15*10^{-47}$	$.86*10^{-48}$	$.81*10^{-48}$	$.63*10^{-48}$
4	$.26*10^{-47}$	$.14*10^{-47}$	$.21*10^{-47}$	$.13*10^{-47}$

K=20 D=68

m	x=5.0 η_m
0	$.33*10^{-68}$
1	$.95*10^{-68}$
2	$.88*10^{-68}$
3	$.23*10^{-67}$
4	$.31*10^{-67}$

These are the same values as those generated by ILEG2, the difference being that ILEG2 generates the function values a column at a time and ILEG3 generates them a row at a time. Therefore the relative error of the Q_n^m calculated by ILEG3 can be determined exactly as described above for ILEG2. Table 5 gives the maximum relative error for each row generated by ILEG3,

$$\eta_n = \max_{0 \leq m \leq 4} \eta_n^m$$

for various values of x and K. The largest relative error observed here is also $.47*10^{-27}$ when x=10.0, K=10 and D=28 so the conclusion stated for ILEG2 is also true here: when K is so close to the total number of digits which can be stored in the multiple precision number of length K, there may only be D-1 significant digits in the result.

LEG1 and LEG2 - LEG1 generates to d significant digits the associated Legendre function of the first kind $P_a^m(x)$, for m=0,1,..., mmax given $x \geq 1$ and a where m is an integer and x and a are real, a not an integer.

LEG2 generates to d significant digits the associated Legendre functions of the first kind $P_{a+n}^m(x)$, for n=0,1,...,nmax given $m \geq 0$, $x \geq 1$ and a where m and n are integers and x and a are real, a not an integer.

TABLE 5

ILEG3

K=10 D=28

n	x=1.1 η_n	x=5.0 η_n	x=10.0 η_n	x=25.0 η_n
0	$.12*10^{-27}$	$.92*10^{-28}$	$.35*10^{-27}$	$.40*10^{-27}$
1	$.14*10^{-27}$	$.11*10^{-27}$	$.40*10^{-27}$	$.36*10^{-27}$
2	$.13*10^{-27}$	$.64*10^{-28}$	$.45*10^{-27}$	$.39*10^{-27}$
3	$.13*10^{-27}$	$.78*10^{-28}$	$.47*10^{-27}$	$.40*10^{-27}$
4	$.85*10^{-28}$	$.67*10^{-28}$	$.47*10^{-27}$	$.43*10^{-27}$

K=15 D=48

n	x=1.1 η_n	x=5.0 η_n	x=10.0 η_n	x=25.0 η_n
0	$.19*10^{-47}$	$.13*10^{-47}$	$.12*10^{-47}$	$.40*10^{-47}$
1	$.15*10^{-47}$	$.11*10^{-47}$	$.14*10^{-47}$	$.38*10^{-47}$
2	$.18*10^{-47}$	$.67*10^{-48}$	$.12*10^{-47}$	$.37*10^{-47}$
3	$.14*10^{-47}$	$.42*10^{-48}$	$.74*10^{-48}$	$.39*10^{-47}$
4	$.14*10^{-47}$	$.47*10^{-48}$	$.23*10^{-48}$	$.43*10^{-47}$

K=20 D=68

n	x=5.0 η_n
0	$.31*10^{-67}$
1	$.17*10^{-67}$
2	$.14*10^{-67}$
3	$.13*10^{-67}$
4	$.12*10^{-67}$

For these cases of $P_a^m(x)$ where a is not an integer or restricted to a half-integer an independent calculation is very difficult since there are no explicit expressions available and the infinite series representation is extremely complicated and slow to converge. It is for this very reason that Gautschi's algorithm is so useful since it does not require that any values of the function be known in advance. In fact this author found no previous table computations made for a other than integer or half-integer values. It does however make the testing of these programs difficult. Zhurina and Karmazina[2] have recorded, as part of their work on the conical functions, the formulae for $P_{-1/2+i\tau}(x)$ and $P^1_{-1/2+i\tau}(x)$, valid for $|1-x|<2$. These were used with $\tau = 0$ in a program which stored the multiple precision numbers in arrays of length K+2. The results, $\{P_{-1/2}(x)\}^*$ and $\{P^1_{-1/2}(x)\}^*$, truncated to multiple precision numbers of length K were then considered exact. Test 1 of Table 6 shows the relative errors

$$\eta 1 = \left| \frac{\{P^m_{-1/2}\}^* - \{P^m_{-1/2}\}^1}{\{P^m_{-1/2}\}^*} \right|, \quad m=0 \text{ or } 1$$

and

$$\eta 2 = \left| \frac{\{P^m_{-1/2}\}^* - \{P^m_{-1/2}\}^2}{\{P^m_{-1/2}\}^*} \right|, \quad m=0 \text{ or } 1$$

of $\{P^m_{-1/2}\}^1$ and $\{P^m_{-1/2}\}^2$ generated by LEG1 and LEG2 respectively. In the case of LEG2 which uses LEG1 to obtain values of P^m_a and P^m_{a+1}, the

calling program specified a = -2.5 and n = 2 and then tested the relative error of P^m_{a+2} so that we would not merely be retesting values generated by LEG1 as would be the case if a = -.5 and n=0 was specified. The maximum relative error observed was $.34*10^{-26}$ for LEG2 with x=2.5, K=10 and D=25. This indicates that there were 25 significant digits generated as specified by the calling program.

To check other values of m and a, an "internal consistency" test was run. First a 5x5 matrix was generated from both LEG1 and LEG2,

$$\begin{bmatrix} \{P^0_{a_0}\}^i & \{P^0_{a_0+1}\}^i & \cdots & \{P^0_{a_0+4}\}^i \\ \{P^1_{a_0}\}^i & & & \vdots \\ \vdots & & & \\ \{P^4_{a_0}\}^i & \cdots & & \{P^4_{a_0+4}\}^i \end{bmatrix}$$

where i=1 or 2 for values of LEG1 or LEG2 respectively, and then the relative difference at each point was determined

$$\eta^m_a = \left| \frac{\{P^m_a\}^1 - \{P^m_a\}^2}{\{P^m_a\}^1} \right| , \quad 0 \leq m \leq 4, \; a_0 \leq a \leq a_0+4.$$

Test 2 of Table 6 gives the maximum relative difference for each row

$$\eta_m = \max_{a_0 \leq a \leq a_0+4} \eta^m_a .$$

TABLE 6

LEG1 and LEG2

Test 1

K=10 D=25

	x=1.5		x=2.0		x=2.5	
m	$\eta 1$	$\eta 2$	$\eta 1$	$\eta 2$	$\eta 1$	$\eta 2$
0	$.63*10^{-28}$	$.71*10^{-28}$	$.27*10^{-27}$	$.34*10^{-27}$	$.40*10^{-27}$	$.53*10^{-27}$
1	$.11*10^{-28}$	$.91*10^{-27}$	$.23*10^{-28}$	$.12*10^{-26}$	$.78*10^{-28}$	$.34*10^{-26}$

K=15 D=45

	x=1.5		x=2.0		x=2.5	
m	$\eta 1$	$\eta 2$	$\eta 1$	$\eta 2$	$\eta 1$	$\eta 2$
0	$.56*10^{-48}$	$.63*10^{-48}$	$.14*10^{-48}$	$.17*10^{-48}$	$.13*10^{-47}$	$.18*10^{-47}$
1	$.77*10^{-49}$	$.65*10^{-47}$	$.12*10^{-48}$	$.66*10^{-47}$	$.13*10^{-48}$	$.56*10^{-47}$

Test 2

K=10 D=25

x=1.5

	$a_0 = -1/2$	$a_0 = -1/3$
m	η_m	η_m
0	$.34*10^{-27}$	$.47*10^{-27}$
1	$.34*10^{-27}$	$.31*10^{-27}$
2	$.38*10^{-27}$	$.26*10^{-27}$
3	$.51*10^{-27}$	$.27*10^{-27}$
4	$.48*10^{-27}$	$.23*10^{-27}$

K=15 D=45

x=1.5

$a_0 = -1/2$

m	η_m
0	$.26*10^{-47}$
1	$.36*10^{-47}$
2	$.37*10^{-47}$
3	$.40*10^{-47}$
4	$.30*10^{-47}$

These results also verify that there were at least D significant digits as specified in the calling program.

CONIC - CONIC generates to d significant digits Mehler's conical functions $P^{m}_{-1/2+i\tau}(x)$, for m=0,1,...,mmax given $x \geq 1$ and τ where m is an integer and x and τ are real.

Exact values, $\{P_{-1/2+i\tau}(x)\}^{*}$ and $\{P^{1}_{-1/2+i\tau}(x)\}^{*}$, were obtained from the formulae of Zhurina and Karmazina, as described above under LEG1 and LEG2, for testing values of $P_{-1/2+i\tau}(x)$ and $P^{1}_{-1/2+i\tau}(x)$ generated by CONIC. Test 1 of Table 7 shows the maximum relative error observed for each combination of x and τ,

$$\eta_{max} = \max_{0 \leq m \leq 1} \left| \frac{\{P^{m}_{-1/2+i\tau}\}^{*} - P^{m}_{-1/2+i\tau}}{\{P^{m}_{-1/2+i\tau}\}^{*}} \right| .$$

The maximum relative error observed is $.80*10^{-44}$ when x=2.0, τ=0.0, K=15 and D=43. This indicates that there were D significant digits as specified by the calling program.

To check other values of m, an internal consistency test was run comparing the values $P^{m}_{-1/2+i*0}(x)$ generated by CONIC with the values $\{P^{m}_{-1/2}(x)\}^{*}$ generated by LEG1 for $0 \leq m \leq 4$. Test 2 of Table 7 gives the maximum relative difference

$$\eta_{max} = \max_{0 \leq m \leq 4} \left| \frac{\{P^{m}_{-1/2}\}^{*} - P^{m}_{-1/2+i*0}}{\{P^{m}_{-1/2}\}^{*}} \right| .$$

TABLE 7

CONIC

Test 1

K=10 D=23

	x=1.5	x=2.0	x=2.5
τ	η_{max}	η_{max}	η_{max}
0.0	$.77*10^{-26}$	$.47*10^{-24}$	$.57*10^{-24}$
15.0	$.25*10^{-25}$	$.48*10^{-25}$	$.73*10^{-24}$

K=15 D=43

	x=1.5	x=2.0
τ	η_{max}	η_{max}
0.0	$.57*10^{-44}$	$.80*10^{-44}$
15.0	$.28*10^{-46}$	

Test 2

K=10 D=23

x=1.5 $\eta_{max} = .18*10^{-25}$

K=15 D=43

x=1.5 $\eta_{max} = .69*10^{-44}$

The maximum relative difference observed was $.69*10^{-44}$ when x = 1.5, K=15 and D=43 again verifying that there were D significant digits generated.

TOROID - TOROID generates to d significant digits the toroidal functions of the second kind $Q^m_{-1/2+n}(x)$, for n=0,1,...,nmax given x > 1 and m where m and n are integers and x is real.

The testing of TOROID follows the same pattern as that of ILEG1: determining the error of key elements in a matrix of values generated by TOROID and then finding an approximation to the errors of the other elements by means of the recurrence relation (4-1). A 5x6 matrix was chosen for study ($0 \leq m \leq 4$, $0 \leq n \leq 5$).

$$\begin{bmatrix} Q^0_{-1/2} & Q^0_{1/2} & \cdots & Q^0_{4\ 1/2} \\ Q^1_{-1/2} & & & Q^1_{4\ 1/2} \\ \vdots & & & \\ Q^4_{-1/2} & Q^4_{1/2} & & Q^4_{4\ 1/2} \end{bmatrix}$$

For an independent calculation, the infinite series representation

$$Q^4_{-1/2+n}(x) = \frac{1\cdot 3\cdot 5\cdot \ldots \cdot (2n+7)\cdot \pi}{2^{4+1/2+n}\,(x^2-1)^{1/4}(x+\sqrt{x^2-1})^n} * \sum_{k=0}^{\infty} \frac{\prod_{i=1}^{k} (4+\frac{1}{2}+i)(\frac{1}{2}-4+i)}{k!(n+k)!}(-t)^k$$

where $t = \dfrac{x - \sqrt{x^2-1}}{2\sqrt{x^2-1}}$,

derived from a formula[3] for $Q_n^m(x)$ valid for $|x| > 1$, was used in a program which stored the multiple precision numbers in arrays of length K+2. The results, $\{Q^4_{-1/2+n}\}^*$, n=0,...,4,truncated to multiple precision numbers of length K were then considered exact and used to determine the absolute error

$$\varepsilon^4_{-1/2+n} = |\{Q^4_{-1/2+n}\}^* - Q^4_{-1/2+n}| \ , \ n=0,\ldots,4,$$

of the $Q^4_{-1/2+n}$ generated by TOROID.

The relationship of the errors in the other elements to the errors of the elements in the 5th row, arrived at as explained in the section about ILEG1, is given by the following table.

$$\begin{array}{llllll}
\frac{2^4x}{5\cdot3}\frac{(x^2-1)^2}{x^4}\varepsilon^4_{\frac12} & \frac{2^4}{5\cdot3}\frac{(x^2-1)^2}{x^4}\varepsilon^4_{\frac12} & \frac{2^4}{3\cdot3}\frac{(x^2-1)^2}{x^4}\varepsilon^4_{\frac32} & \cdots & \frac{2^4}{9\cdot7\cdot5\cdot3}\frac{(x^2-1)^2}{x^4}\varepsilon^4_{\frac92} \\
\frac{2^3x}{5\cdot3^2}\frac{(x^2-1)^{\frac32}}{x^3}\varepsilon^4_{\frac12} & \frac{2^3}{5\cdot3}\frac{(x^2-1)^{\frac32}}{x^3}\varepsilon^4_{\frac12} & \frac{2^3}{3}\frac{(x^2-1)^{\frac32}}{x^3}\varepsilon^4_{\frac32} & \cdots & \frac{2^3}{7\cdot5\cdot3}\frac{(x^2-1)^{\frac32}}{x^3}\varepsilon^4_{\frac92} \\
\frac{2^2x}{5^2}\frac{(x^2-1)}{x^2}\varepsilon^4_{\frac12} & \frac{2^2}{5\cdot3}\frac{(x^2-1)}{x^2}\varepsilon^4_{\frac12} & \frac{2^2}{3}\frac{(x^2-1)}{x^2}\varepsilon^4_{\frac32} & \cdots & \frac{2^2}{5\cdot3}\frac{(x^2-1)}{x^2}\varepsilon^4_{\frac92} \\
\frac{2x}{7}\frac{(x^2-1)^{\frac12}}{x}\varepsilon^4_{\frac12} & \frac{2}{5}\frac{(x^2-1)^{\frac12}}{x}\varepsilon^4_{\frac12} & \frac{2}{3}\frac{(x^2-1)^{\frac12}}{x}\varepsilon^4_{\frac32} & \cdots & \frac{2}{3}\frac{(x^2-1)^{\frac12}}{x}\varepsilon^4_{\frac92} \\
\varepsilon^4_{-\frac12} & \varepsilon^4_{\frac12} & \varepsilon^4_{\frac32} & \cdots & \varepsilon^4_{\frac92}
\end{array}$$

TABLE 8

TOROID

K=10 D=25

m	x=1.5 η_m	x=3.0 η_m	x=10.0 η_m
0	$.16*10^{-25}$	$.81*10^{-25}$	$.97*10^{-25}$
1	$.15*10^{-25}$	$.61*10^{-25}$	$.70*10^{-25}$
2	$.59*10^{-26}$	$.21*10^{-25}$	$.23*10^{-25}$
3	$.60*10^{-27}$	$.20*10^{-26}$	$.32*10^{-26}$
4	$.47*10^{-28}$	$.15*10^{-27}$	$.32*10^{-27}$
	r_{max}	r_{max}	r_{max}
	$.34*10^{-27}$	$.22*10^{-27}$	$.29*10^{-27}$

K=15 D=45

m	x=1.5 η_m	x=3.0 η_m	x=10.0 η_m
0	$.82*10^{-46}$	$.41*10^{-46}$	$.28*10^{-45}$
1	$.75*10^{-46}$	$.30*10^{-45}$	$.18*10^{-45}$
2	$.30*10^{-46}$	$.11*10^{-45}$	$.60*10^{-46}$
3	$.30*10^{-47}$	$.98*10^{-47}$	$.11*10^{-46}$
4	$.49*10^{-48}$	$.92*10^{-48}$	$.80*10^{-48}$
	r_{max}	r_{max}	r_{max}
	$.37*10^{-47}$	$.18*10^{-47}$	$.48*10^{-47}$

K=20 D=65

m	x=3.0 η_m
0	$.81*10^{-65}$
1	$.61*10^{-65}$
2	$.21*10^{-65}$
3	$.20*10^{-66}$
4	$.15*10^{-67}$
	r_{max}
	$.25*10^{-67}$

Table 8 gives the maximum relative error for each row generated by TOROID

$$\eta_m = \max_{0 \leq n \leq 5} \frac{\varepsilon^m_{-1/2+n}}{\left| Q^m_{-1/2+n} \right|}$$

where the $\varepsilon^m_{-1/2+n}$ are calculated from the above table. The maximum relative residual

$$r_{max} = \max_{\substack{1 \leq n \leq 5 \\ 0 \leq m \leq 3}} \frac{\left| Q^{m+1}_{-1/2+n} - \frac{1}{\sqrt{x^2-1}} \left[\left(-\frac{1}{2} + n - m\right) x Q^m_{-1/2+n} - \left(-\frac{1}{2} + n + m\right) Q^m_{-1/2+n-1} \right] \right|}{\left| Q^{m+1}_{-1/2+n} \right|}$$

was calculated using the values $Q^m_{-1/2+n}(x)$ generated by TOROID. The results verify that there were D significant digits as specified by the calling program.

FOOTNOTES

1. D. Suschowk, "Explicit Formulae for 25 of the Associated Legendre Functions of the Second Kind," MTAC 13 (1959), pp. 303-305.

2. M. I. Zhurina and L. N. Karmazina, Tables and Formulae for the Spherical Functions $P^m_{-1/2+i\tau}(z)$ (Pergamon Press, New York, 1966), p. 11.

3. NBS Mathematical Tables Project, p. xix.

V. SUMMARY AND SUGGESTIONS FOR FURTHER STUDY

From the results given in the preceeding chapter, we can conclude that the goal of obtaining Legendre functions in multiple precision has been reached to the extent that there is a group of FORTRAN programs available which can be used on an IBM 360, and probably with only slight modifications on any other computer which accepts FORTRAN. The value of these multiple precision programs, even when only single precision results are desired, is especially pointed out by the subroutine CONIC where accumulation of round-off error seriously affects the convergence of the algorithm. With these programs you can merely increase the length of the multiple precision numbers, while still asking for the same number of significant digits and thus eliminate the effect of the error accumulation by "discarding" the non-significant digits at the end of the calculation.

Perhaps of more importance than the Legendre function programs is the fact that there are now available a group of basic subroutines which can be used to calculate many functions in multiple precision.

It was mentioned in the introduction that a good solution to the problem of tables would be a storage system on magnetic tape. This is particularly true in the case of multiple precision numbers since the computation of a single function value can use up a large amount of computer time. Since this table would be generated on a specific computer and hopefully be stored in such a way as to be useable on many, and since filling a table would require computation

of the function at many different points, it would probably be desirable to rewrite some or all of the basic subroutines in assembly language to attempt to decrease the total execution time.

In the case of the Legendre function programs given here there is another possible way to save execution time which should be investigated before tables are generated from these programs. These programs are very good because they do not require starting values to be known in advance as most programs do since functions are often generated from their recurrence relations. However they do use an iterative procedure which can consume a great deal of time. Since these programs generate a column of function values at a time, maximum efficiency could probably be attained by generating the first two columns of a table from these programs and then using a recurrence relation across the rows to fill in the rest of the table. It might be necessary to also generate every ith column of the table from these programs to prevent error growth. Some work would be required to find the best combination in terms of time and accuracy. It should be remembered that the estimation of where to start the iterative procedure was based on 50 elements in a column being generated at one time (nmax=50) so that overall time could also be reduced by using this value.

BIBLIOGRAPHY

1. Abramowitz, M. and Stegun, I.A., ed., Handbook of Mathematical Functions, (Dover, New York, 1965).

2. Bateman, H., Higher Transcendental Functions, Vol. 1, (McGraw-Hill, New York, 1953).

3. Davis, H. T., Tables of Higher Mathematical Functions, Vol. 2, (Bloomington, 1935).

4. Filho, A. M. S. and Schwachheim, G., "Algorithm 309 Gamma Function with Arbitrary Precision," Comm. Assoc. Comp. Mach. 10 (1967), pp. 511-512.

5. Gautschi, W., Strength and Weakness of Three-Term Recurrence Relations, (Unpublished).

6. Gautschi, W., "Algorithm 229 Legendre Functions for Arguments Larger than One," Comm. Assoc. Comp. Mach. 8 (1965), pp. 488-492.

7. Hill, I. D., "Algorithm 34 Procedures for the Basic Arithmetical Operations in Multiple-Length Working," Computer J. 11 (1968), pp. 232-235.

8. Hobson, E. W., The Theory of Spherical and Ellipsoidal Harmonics, (University Press, Cambridge, 1931).

9. NBS Mathematical Tables Project, Tables of Associated Legendre Functions, (Columbia University Press, New York, 1945).

10. "A New Approximation to π," MTAC 2 (1947), pp. 245-248.

11. "Review 275," MTAC 2 (1946), pp. 68-69.

12. Rotenberg, A., "The Calculation of Toroidal Harmonics," Math. Comput. 14 (1960), pp. 274-276.

13. Suschowk, D., "Explicit Formulae for 25 of the Associated Legendre Functions of the Second Kind," MTAC 13 (1959), pp. 303-305.

14. Uhler, H. S., "A New Table of Reciprocals of Factorials and Some Derived Numbers," Trans. Conn. Acad. Arts. Sci. 32 (1937), pp. 381-434.

15. Uhler, H. S.,"Log π and other Basic Constants," Proc. Nat. Acad. Sci. 24 (1938), pp. 23-30.

16. Uhler, H. S., "Recalculation and Extension of the Modulus and of the Logarithms of 2, 3, 5, 7 and 17," Proc. Nat. Acad. Sci. 26 (1940), pp. 205-212.

17. Uhler, H. S., "The Coefficients of Sterling's Series for Log $\Gamma(x)$," Proc. Nat. Acad. Sci. 28 (1942), pp. 59-61.

18. Uhler, H. S., Original Tables to 137 Decimal Places of Natural Logarithms for Factors of the Form $1\pm n\cdot 10^{-p}$, Enhanced by Auxiliary Tables of Logarithms of Small Integers, (New Haven, Connecticut, 1942).

19. Uhler, H. S., "Natural Logarithms of Small Prime Numbers," Proc. Nat. Acad. Sci. 29 (1943), pp. 319-325.

20. Werner, H. and Collinge, R., "Chebyshev Approximations to the Gamma Function," Math. Comput. 15 (1961), pp. 195-197.

21. Zhurina, M. I. and Karmazina, L. N., Tables and Formulae for the Spherical Functions $P^m_{-1/2+i\tau}(z)$, Mathematical Tables Series Vol.40, (Pergamon Press, New York, 1966).

APPENDIX

ENTRY NAMES
LNGCNS
COPY
SET
LNGTHN
SHORTN
ADD
SBTRCT
MLYPLY
MLTINT
RECIP
DIVIDE

IDENTIFICATION *** FORTRAN Package for Multiple Precision Basic Arithmetic

PURPOSE *** This is a group of subroutines which perform arithmetic operations on multiple precision numbers. Each multiple precision number is stored in an integer array of arbitrary length K. The multiple precision number X is stored in the form X.EQ.A*(10**4)**B where 1.LE.A.LT.10**4 as follows:

```
INTEGER*4 X(K)

X(1) = sign of the number: +1 if positive
                           -1 if negative
                            0 if zero (In this case, X(2) to
                              X(K) are disregarded)

X(2) = exponent B of 10**4

X(3) = integral part of the mantissa, 1.LE.X(3).LT.10**4

X(4) to X(K) = fractional part of the mantissa, stored 4
               decimal digits per location
```

X has at least (K-3)*4+1 significant decimal digits. K must be less than or equal to 50 so the maximum number of decimal digits which can be stored is 189. The maximum magnitude of

X is 9.999...*10**8589934591. The operations are performed in floating point decimal arithmetic.

OTHER SUBROUTINES USED *** FIXPI and FRXPI by LNGCNS and FRXPI and UNDERZ by SHORTN. ADJUST and READJT are subroutines used internally by ADD, SBTRCT and MLTINT. Also many of the subroutines call other subroutines in the package.

USAGE ***

1) To initialize constants needed by other subroutines in the package:

```
INTEGER K
:
CALL LNGCNS(K,4,8,75)
```

where the parameters in the calling sequence are:

K - The number of locations of the integer arrays which will be used to store the multiple precision numbers. K.LE.50
input

4 - The integer 4 is the number of decimal digits to be held in each register of the integer array, chosen as large as possible with the restriction that 10**(2*4) does not cause integer overflow.
input

8 - The integer 8 is the number of significant decimal digits in a REAL*4 number, rounded up to the next higher integer.
input

75 - The integer 75 is the maximum allowable decimal exponent of a real number rounded down to the next smaller number.
input

NOTE - This subroutine must be called before any of the other subroutines in the package are called.

2) To copy a multiple precision number into the array of another multiple precision number:

```
INTEGER A(KX),B(KX),K
  .
  .
  .
CALL COPY(A,B,K)
```

where the parameters in the calling sequence are:

A - The multiple precision number to be copied.

input

B - On output B=A, where B is a multiple precision number.

output

K - The array length of the multiple precision numbers A and B. K.LE.KX.LE.50

input

3) To set a multiple precision number equal to an integer:

```
INTEGER A(KX),I,K
  .
  .
  .
CALL SET(A,I,&N,K)
```

where the parameters in the calling sequence are:

A - On output A=I, where A is a multiple precision number.

output

I - The integer which A is to be set equal to.

input

N - Control is returned to the statement labeled N(an actual integer) if the array A is not long enough to hold the integer I.

input

K - The array length of the multiple precision number

A. K.LE.KX.LE.50

input

4) To set a multiple precision number equal to a REAL*4 number:

```
INTEGER A(KX),K
REAL X
.
.
.
CALL LNGTHN(X,A,K)
```

where the parameters in the calling sequence are:

X - The real number to which A will be set equal.

input

A - On output A=X, where A is a multiple precision number.

output

K - The array length of the multiple precision number A.

K.LE.KX.LE.50

input

5) To obtain the REAL*4 representation of a multiple precision number:

```
INTEGER A(KX),K
REAL X
.
.
.
CALL SHORTN(A,X,&N,K)
```

where the parameters in the calling sequence are:

A - The multiple precision number.

input

X - On output X contains the ordinary REAL*4 representation of A.

output

N - Control is returned to the statement labeled N(an actual integer) if overflow would be produced.

input

K - The array length of the multiple precision number A. K.LE.KX.LE.50

input

6) To add two multiple precision numbers:

```
INTEGER A(KX),B(KX),C(KX),K
  .
  .
  .
CALL ADD(A,B,C,K)
```

where the parameters in the calling sequence are:

A, B - The multiple precision numbers to be added.

input

C - On output C=A+B where C is a multiple precision number.

output

K - The array length of the multiple precision numbers A, B and C. K.LE.KX.LE.50

input

7) To subtract two multiple precision numbers:

```
INTEGER A(KX),B(KX),C(KX),K
  .
  .
  .
CALL SBTRCT(A,B,C,K)
```

where the parameters in the calling sequence are:

A,B - The multiple precision number B is to be subtracted from the multiple precision number A.

input

C - On output C=A-B where C is a multiple precision number.

output

K - The array length of the multiple precision numbers, A, B and C. K.LE.KX.LE.50

input

8) To multiply two multiple precision numbers:

```
INTEGER A(KX),B(KX),C(KX),K
  :
CALL MLTPLY(A,B,C,K)
```

where the parameters in the calling sequence are:

A,B - The multiple precision numbers to be multiplied.

input

C - On output C=A*B where C is a multiple precision number.

output

K - The array length of the multiple precision numbers A, B and C. K.LE.KX.LE.50

input

9) To multiply a multiple precision number by an integer:

```
INTEGER A(KX),B(KX),I,K
  :
CALL MLTINT(A,I,B,&N,K)
```

where the parameters in the calling sequence are:

A - The multiple precision number.

input

I - The integer by which A is to be multiplied. ABS(I).LT.10**4

input

B - On output B=A*I where B is a multiple precision number.

output

N - Control is returned to the statement labeled N(an actual integer) if ABS(I).GE.10**4.

input

K - The array length of the multiple precision numbers A and B. K.LE.KX.LE.50

input

NOTE - This subroutine is much faster than MLTPLY and should be used whenever possible.

10) To find the reciprocal of a multiple precision number:

```
INTEGER A(KX),B(KX),K
  .
  .
  .
CALL RECIP(A,B,K)
```

where the parameters in the calling sequence are:

A - The multiple precision number for which the reciprocal is desired. A.NE.0

input

B - On output B=1/A where B is a multiple precision number. If A.EQ.0 the ordinary system failure for division by zero will occur.

output

K - The array length of the multiple precision numbers A and B. K.LE.KX.LE.50

input

11) To divide two multiple precision numbers:

```
INTEGER A(KX),B(KX),C(KX),K
:
CALL DIVIDE (A,B,C,K)
```

where the parameters in the calling sequence are:

A,B - The multiple precision number A is to be divided by the multiple precision number B. B.NE.0

input

C - On output C=A/B where C is a multiple precision number. If B.EQ.0 the ordinary system failure for division by zero will occur.

output

K - The array length of the multiple precision numbers A, B and C. K.LE.KX.LE.50

input

ACCURACY ***

COPY - no loss of accuracy

SET - "

LNGTHN - the result has REAL*4 accuracy

SHORTN - "

ADD - the error is at most 1 in the least significant digit

SBTRCT - "

MLTPLY - the error is at most 1 in the least significant digit

MLTINT - "

RECIP - the error is at most 1 in the ((K-3)*4+1)st digit

DIVIDE - "

EXECUTION TIME *** Using the FORTRAN G compiler the execution time in seconds is given approximately by (A+BK+CK**2)*10**(-4) where K is the length of the arrays used to store the multiple precision numbers.

	A	B	C
LNGCNS	2		
COPY	.55	.095	
SET	1.5	.072	
LNGTHN	2.5	.094	
SHORTN	3		
ADD	3	.69	
SBTRCT	4.2	.81	
MLTPLY	-4.8	-.23	.45
MLTINT	1.1	.35	
RECIP	-48	-34	6.2
DIVIDE	-52	-34	6.7

```
      SUBROUTINE LNGCNS(KK,NN,WW,EE)
C        THIS IS THE FIRST SUBROUTINE OF A FORTRAN PACKAGE TO DO
C     MULTIPLE PRECISION ARITHMETIC.  THE PACKAGE IS A TRANSLATION BY
C     HENRY C. THACHER, JR. OF ALGORITHM 34, COMPUTER J. 11(1968)232-235
C     BY I. D. HILL.
C        EACH NUMBER IS STORED IN AN INTEGER ARRAY WITH N DIGITS PER
C     REGISTER, WHERE N IS AT CHOICE(BUT 10**2*N MUST NOT LEAD TO INTEG-
C     ER OVERFLOW).  THE NUMBER IS STANDARDIZED SO THAT THE MANTISSA IS
C     LESS THAN 10**N, BUT GREATER THAN OR EQUAL TO 1.
C        ARRAY ELEMENT 1 (0 IN THE ALGOL) HOLDS THE SIGN OF THE NUMBER,
C     +1 IF THE NUMBER IS POSITIVE, -1 IF NEGATIVE, OR 0 IF ZERO.  IF
C     THE ELEMENT IS ZERO, THE REST OF THE ARRAY ELEMENTS ARE IRRELEVANT
C     -- THEY MAY HOLD VALUES, BUT THESE ARE MEANINGLESS.
C        ARRAY ELEMENT 2 HOLDS THE EXPONENT IN ORDINARY INTEGER FORM.
C     IT IS THE EXPONENT OF 10**N, AND MAY TAKE ANY VALUE NORMALLY ALLO-
C     WED FOR INTEGERS.
C        ARRAY ELEMENT 3 HOLDS THE INTEGRAL PART OF THE MANTISSA, AND
C     THE REMAINING ELEMENTS HOLD THE FRACTIONAL PART, N DECIMAL DIGITS
C     TO EACH ELEMENT IN TURN.
C        IF THE ARRAY HAS BOUNDS (1,K) THE NUMBER IS REPRESENTED TO K-2
C     SIGNIFICANT DIGITS IN THE SCALE OF 10**N, I.E. AT LEAST N(K-3)+1
C     SIGNIFICANT DECIMAL FIGURES.
C        CERTAIN CONSTANTS APPEAR IN A NUMBER OF THE PROCEDURES, AND
C     SOME OF THE PROCEDURES CALL EACHOTHER. FOR THE SAKE OF SIMPLICITY,
C     THESE CONSTANTS ARE STORED IN COMMON LABELLED LONG, EXCEPT FOR THE
C     ARRAY LENGTH, K.   THE IDENTIFIERS THAT MUST BE INCLUDED ARE
C     INTEGER N,W,T,S,EX AND REAL R,TR,RM.  THEY MUST BE ASSIGNED VALUES
C     BY CALLING THE SUBROUTINE LNGCNS BEFORE ANY OTHER LONG OPERATIONS
C     ARE USED.
      COMMON/LONG/N,W,T,S,R,TR,EX,RM
      INTEGER K,N,W,T,S,EX
      REAL R, TR,RM
      INTEGER KK,WW,NN,EE
C        KK GIVES THE SIZE OF INTEGER ARRAYS TO BE USED TO HOLD LONG
C     NUMBERS. NN GIVES THE NUMBER OF DECIMAL DIGITS TO BE HELD IN EACH
C     REGISTER, EXCEPT WHERE THERE IS A SPECIAL REASON FOR CHOOSING SOME
C     OTHER VALUE, NN SHOULD BE AS LARGE AS POSSIBLE, SUBJECT TO THE
C     CONDITION THAT 10**(2*NN) MUST BE AN ALLOWABLEINTEGER IN THE ARITH
C     METIC BEING USED.  WW GIVES THE NUMBER OF SIGNIFICANT DECIMAL FIG-
C     URES IN THE REPRESENTATION OF A REAL NUMBER IN THE ARITHMETIC BEIN
C     USED.  WHERE THE MACHINE WORKS IN BINARY MODE, THERE IS NO EXACT
C     INTEGER CORRESPONDING TO THIS.  THE VALUE SHOULD BE ROUNDED UP TO
C     THE NEXT INTEGER. EE GIVES THE MAXIMUM ALLOWABLE DECIMAL EXPONENT
C     OF A REAL NUMBER.  WHERE THE MACHINE WORKS IN BINARY MODE, THERE
C     IS NO EXACT INTEGER CORRESPONDING TO THIS.  THE VALUE SHOULD BE
C     ROUNDED DOWN TO THE NEXT SMALLER INTEGER.
      K = KK
      N = NN
      W = WW
      T = 10**N
      TR = T
      S = T/2
      R = 1.0/TR
      EX = EE
      RM = 10.0**(-EX)
      RETURN
      END
```

```
      SUBROUTINE COPY(A,B,K)
      INTEGER A(1),B(1)
C        THE PROCEDURE IS THE LONG EQUIVALENT OF B = A.
      COMMON/LONG/N,W,T,S,R,TR,EX,RM
      REAL R, TR,RM
      INTEGER   N,W,T,S,EX
      DO 5 J = 1,K
    5 B(J) = A(J)
      RETURN
      END
      SUBROUTINE SET(A,I,*,K)
      INTEGER A(1),I
C        THE PROCEDURE IS THE LONG EQUIVALENT OF A = I. IT DIFFERS FROM
C     THE SUBROUTINE COPY IN THAT I IS AN INTEGER, AND NOT AN INTEGER
C     ARRAY.
C        IF THE ARRAY A IS NOT LONG ENOUGH TO HOLD THE INTEGER I, A
C     RETURN TO THE LABEL PARAMETER OCCURS.
      COMMON/LONG/N,W,T,S,R,TR,EX,RM
      REAL R, TR,RM
      INTEGER   N,W,T,S,EX
      INTEGER J,IL,P,Q
      A(1) = -1
      IL = -I
      IF(I) 1,2,3
    3 A(1) = 0
      IL = -IL
    2 A(1) = A(1) + 1
      IF(I.EQ.0) RETURN
    1 P = IL
      J = 3
    5 P = P/T
      IF(P) 6,6,7
    7 J = J + 1
C     PROCEDURE SET IN THAT X IS REAL.  IT IS IMPORTANT TO REALIZE THAT
C        THE PROCEDURE IS THE LONG EQUIVALENT OF A = X. IT DIFFERS FROM
      REAL X
      INTEGER A(1)
      SUBROUTINE LNGTHN(X,A,K)
      END
      RETURN
   17 A(2) = J - 3
   15 A(P) = 0
   16 DO 15 P = IL,K
      IF(IL - K) 16,16,17
   11 IL = J + 1
      GO TO 10
      P = P - 1
      IL = Q
      A(P) = IL - T*Q
    9 Q = IL/T
   10 IF(P-2) 11,11,9
      P = J
    6 IF(J.GT.K) RETURN 1
      GO TO 5
C     THE RESULTING ACCURACY CANNOT BE ANY GREATER THAN THAT OF THE OR-
C     DINARY REAL REPRESENTATION.
      COMMON/LONG/N,W,T,S,R,TR,EX,RM
```

```
      INTEGER   N,W,T,S,EX
      REAL R, TR,RM
      REAL Y,Z,XL,XT
      INTEGER J,M,V
      M = -1
      XL = -X
      IF(XL) 3,2,1
    3 M = 0
      XL = -XL
    2 M = M + 1
    1 A(1) = M
      IF(M) 5,35,5
    5 Y = 1.0
      J = 0
    6 IF(XL - TR)7,8,8
    8 XL = R*XL
      J = J + 1
      GO TO 6
    7 IF(XL - 1.0) 9,10,10
    9 XL = TR*XL
      J = J - 1
      GO TO 7
   10 A(2) = J
      V = W/N + 3
      IF(V.GT.K) V = K
      DO 15 M = 3,V
      J = XL
      A(M) = J
      XT = J
   15 XL = T*(XL - XT)
      M = V + 1
   16 M = M - 1
      IF(M.EQ.3.OR.A(M).LT.T) GO TO 20
      A(M) = 0
      A(M-1) = A(M-1) + 1
      GO TO 16
   20 IF(A(3).LT.T) GO TO 25
      A(3) = 1
      A(2) = A(2) + 1
   25 L = V + 1
      DO 30 M = L,K
   30 A(M) = 0
   35 RETURN
      END
      SUBROUTINE SHORTN(A,X,*,K)
      INTEGER A(1)
C        THE RESULT IS THE ORDINARY REAL REPRESENTATIONOF THE NUMBER
C     HELD IN THE ARRAY A.  RETURN TO THE LABEL PARAMETER OCCURS IF OVER
C     FLOW WOULD BE PRODUCED.
      COMMON/LONG/N,W,T,S,R,TR,EX,RM
      REAL R, TR,RM
      INTEGER   N,W,T,S,EX
      REAL X,Y
      INTEGER J,V
      IF(N*(A(2)+1).GE.EX) RETURN 1
      CALL UNDERZ('OFF')
      V = W/N + 2
```

```
      Y = R
      X = A(3)
      IF(V.GT.K - 1) V = K - 1
      IF(V.LT.4) V=4
      DO 10 J = 4,V
      X = Y*A(J) + X
   10 Y = R*Y
C        THIS LOOKS AS IF IT COULD BE TIDIED UP SOME.
      J = V + 1
      T1 = 0
      IF(J.LT.K .AND. A(J+1).GE.S) T1 = 1
      X = Y*(A(J) + T1) + X
       T1 = TR**A(2)
      IF(T1*RM*X.GT.1.0)GO TO 20
      X = T1*X
      T1 = A(1)
      X=T1*X
      CALL UNDERZ('ON')
      RETURN
   20 CALL UNDERZ('ON')
      RETURN 1
      END
      SUBROUTINE ADJUST(D,H,K)
      INTEGER D(1),H
C        THIS PROCEDURE DESTANDARDIZES A LONG NUMBER READY FOR ADDING OR
C     SUBTRACTING.  IT IS NOT INTENDED THAT IT SHOULD BE CALLED EXCEPT
C     BY PROCEDURE ADD OR PROCEDURESBTRCT.
C        H IS THE DIFFERENCE IN EXPONENTS, I.E. THE NUMBER OF PLACES D
C     IS TO BE SHIFTED RIGHT.
      COMMON/LONG/N,W,T,S,R,TR,EX,RM
      INTEGER   N,W,T,S,EX
      REAL R, TR,RM
      INTEGER J,HL,L,T1,T2
      REAL RT
      HL = H
      D(2) = D(2) + HL
      HL = K - HL
      IF(HL - 2) 1,2,3
    1 D(1) = 0
      RETURN
    2 IF(D(3) - S) 4,5,5
    4 D(1) = 0
      RETURN
    5 D(K) = 1
      L = K - 1
      IF(L-3) 10,6,6
    6 DO 8 J = 3,L
    8 D(J) = 0
      RETURN
    3 RT = 2*D(HL+1)
      T1 = R*RT
      D(K) = D(HL) + T1
      HL = HL - 1
      L = K - 3
      DO 20 T1 = 1,L
C     THIS COULD BE TIDIED UP.
      J = K - T1
```

```
      IF(HL - 2) 15,15,16
   16 D(J) = D(HL)
      HL = HL - 1
      GO TO 20
   15 D(J) = 0
   20 CONTINUE
   10 RETURN
      END
      SUBROUTINE READJT(D,E,K)
      INTEGER E,D(1)
C        THIS PROCEDURE READJUSTS A LONG NUMBER IN CERTAIN CASES WHERE
C     ROUNDING CAUSES A FORM OF OVERFLOW.  IT IS NOT INTENDED THAT IT
C     SHOULD BE CALLED EXCEPT BY PROCEDURE ADD OR PROCEDURE MLTINT.
      COMMON/LONG/N,W,T,S,R,TR,EX,RM
      INTEGER   N,W,T,S,EX
      REAL R, TR,RM
      INTEGER J,EL,L,T1
      REAL DT
      EL = E
      D(2) = D(2) + 1
      DT = 2*D(K)
      D(K) = R*DT
      D(K) = D(K) + D(K-1)
      L = K - 4
      DO 5 T1 = 1,L
      J = K - T1
    5 D(J) = D(J-1)
      D(3) = EL
      J = K + 1
   10 J = J - 1
      IF(D(J).NE.T.OR.J.EQ.3) GO TO 11
      D(J) = 0
      D(J-1) = D(J-1) + 1
      GO TO 10
   11 IF(D(3) - T) 12,13,12
   13 D(2) = D(2) - 1
      D(3) = 1
   12 RETURN
      END
      SUBROUTINE ADD(AP,BP,C,K)
      INTEGER AP(1),BP(1), C(1)
C        THIS PROCEDURE IS THE LONG EQUIVALENT OF C = A + B
      COMMON/LONG/N,W,T,S,R,TR,EX,RM
      INTEGER   N,W,T,S,EX
      REAL R, TR,RM
      INTEGER A(50),B(50)
C        THESE DIMENSIONS LIMIT K TO A MAXIMUM OF 50.
      INTEGER J,E,T1
      CALL COPY(AP,A,K)
      CALL COPY(BP,B,K)
      IF(B(1))2,1,2
    1 CALL COPY(A,C,K)
      RETURN
    2 IF(A(1))4,3,4
    3 CALL COPY(B,C,K)
      RETURN
    4 IF(A(1).EQ.B(1)) GO TO 50
```

```
      IF(A(1)) 10,10,5
    5 B(1) = 1
      CALL SBTRCT(A,B,C,K)
      B(1) = -1
      RETURN
   10 A(1) = 1
      CALL SBTRCT(B,A,C,K)
      A(1) = -1
      RETURN
   50 C(1) = A(1)
      IF(A(2) - B(2)) 25,30,15
   15 CALL ADJUST(B,A(2)-B(2), K)
      IF(B(1)) 30,1,30
   25 CALL ADJUST(A,B(2) - A(2),K)
      IF(A(1)) 30,3,30
   30 C(2) = A(2)
      E = 0
      DO 35 T1 = 3,K
      J = K + 3 - T1
      C(J) = A(J) + B(J) + E
      E = 0
      IF(C(J) .LT. T) GO TO 35
      E = 1
      C(J) = C(J) - T
   35 CONTINUE
      IF(E.EQ.1) CALL READJT(C,E,K)
      RETURN
      END
      SUBROUTINE SBTRCT(AP,BP,C,K)
      INTEGER AP(1),BP(1), C(1)
C        THE PROCEDURE IS THE LONG EQUIVALENT OF C = A - B.
      COMMON/LONG/N,W,T,S,R,TR,EX,RM
      INTEGER K,N,W,T,S,EX
      REAL R, TR,RM
      INTEGER J,E,JJ,M
      INTEGER A(50),B(50)
C        THIS DIMENSION LIMITS K TO A MAXIMUM OF 50.
      CALL COPY(AP,A,K)
      CALL COPY(BP,B,K)
      IF(B(1)) 1,2,1
    2 CALL COPY(A,C,K)
C     LABEL 2 IS ALGOL LABEL EE.
      RETURN
    1 IF(A(1)) 4,3,4
    3 B(1) = -B(1)
C     LABEL 3 IS ALGOL LABEL FF
      CALL COPY(B,C,K)
      RETURN
    4 IF(A(1) - B(1)) 5,6,5
    5 JJ = B(1)
      B(1) = A(1)
      CALL ADD(A,B,C,K)
      B(1) = JJ
      RETURN
    6 C(1) = A(1)
      IF(A(2) - B(2)) 7,8,9
    9 CALL ADJUST(B,A(2) - B(2), K)
```

```
      IF(B(1)) 8,2,8
    7 CALL ADJUST(A,B(2) - A(2),K)
      IF(A(1)) 8,3,8
    8 C(2) = A(2)
      E = 0
      DO 20 JJ = 3,K
      J = K + 3 - JJ
      C(J) = A(J) - B(J) + E
      IF(C(J)) 11,12,12
   11 E = -1
      C(J) = C(J) + T
      GO TO 20
   12 E = 0
   20 CONTINUE
      IF(E) 14,21,21
   14 C(1) = -C(1)
      E = K
   18 IF(C(E)) 17,16,17
   16 E = E - 1
      GO TO 18
   17 C(E) = T - C(E)
      J = T - 1
   23 E = E - 1
      IF(E - 3) 21,22,22
   22 C(E) = J - C(E)
      GO TO 23
   21 E = 0
      J = 2
   24 J = J + 1
      IF(C(J)) 25,26,25
   26 E = E + 1
      IF(J-K) 24,28,24
   28 C(1) = 0
      RETURN
   25 IF(E.LE.0) RETURN
      M = K - E
      DO 30 J = 3,M
      I = J + E
   30 C(J) = C(I)
      M = M + 1
      DO 31 J = M,K
   31 C(J) = 0
      C(2) = C(2) - E
      RETURN
      END
      SUBROUTINE MLTPLY(A,B,C,K)
      INTEGER A(1),B(1),C(1)
C        THE PROCEDURE IS THE LONG FQUIVALENT OF C = A*B.
      COMMON/LONG/N,W,T,S,R,TR,EX,RM
      INTEGER   N,W,T,S,EX
      REAL R, TR,RM
      INTEGER G(100),I,J,M,H,L
C        THE MAXIMUM VALUE OF K IS DETERMINED BY THE DIMENSION OF G.
C     WITH THE CURRENT VALUE, K MUST BE NO GREATER THAN 50.
      IF(A(1).NE.0.AND.B(1).NE.0) GO TO 1
      C(1) = 0
      RETURN
```

```
    1 C(1) = A(1)*B(1)
      C(2) = A(2) + B(2)
      L = 2*K - 1
      DO 2 I = 3,L
    2 G(I) = 0
      DO 10 I = 3,K
      DO 10 J = 3,K
      M = I + J - 1
      G(M) = G(M) + A(I)*B(J)
      M = M + 1
    3 M = M - 1
      IF(G(M) - T) 10,4,4
    4 H = G(M)/T
      G(M) = G(M) - T*H
      G(M-1) = G(M-1) + H
      GO TO 3
   10 CONTINUE
      J = K + 1
      IF(G(4).EQ.0) J = J + 1
      IF(J.EQ.3 .OR. K.EQ.3) GO TO 11
      IF(G(J+1).GE.S) G(J) = G(J) + 1
   11 J = J + 1
   12 J = J - 1
      IF(G(J) - T) 16,15,15
   15 G(J) = G(J) - T
      G(J-1) = G(J-1) + 1
      GO TO 12
   16 IF(G(4)) 18,19,18
   19 I = 5
      GO TO 20
   18 C(2) = C(2) + 1
      I = 4
   20 DO 25 J = 3, K
      C(J) = G(I)
   25 I = I + 1
      RETURN
      END
      SUBROUTINE MLTINT(A,I,B,*,K)
      INTEGER A(1),B(1),I
C        THE PROCEDURE IS THE LONG EQUIVALENT OF B = A*I.  IT DIFFERS
C     FROM MLTPLY IN THAT I IS AN INTEGER, AND NOT AN INTEGER ARRAY.  IT
C     IS MUCH FASTER THAN MLTPLY AND SHOULD THEREFORE BE USED IN PREFER-
C     ENCE IF POSSIBLE.  THE ABSOLUTE VALUE OF I MUST BE LESS THAN 10**N
C     A JUMP TO THE LABEL ARGUMENT WILL OCCUR IF THIS CONDITION IS VIO-
C     LATED.
      COMMON/LONG/N,W,T,S,R,TR,EX,RM
      INTEGER   N,W,T,S,EX
      REAL R, TR,RM
      INTEGER J,M,E,IL,JJ
      IL = IABS(I)
      IF(IL.GE.T) RETURN 1
      IF(I) 1,2,3
    2 B(1) = 0
    5 DO 7 J = 2,K
    7 B(J) = 0
      RETURN
    1 B(1) = -A(1)
```

```
      GO TO 4
    3 B(1) = A(1)
    4 IF(B(1).EQ.0) GO TO 5
      E = 0
      B(2) = A(2)
      DO 10 JJ = 3, K
      J = K + 3 - JJ
      M = IL*A(J) + E
      E = M/T
   10 B(J) = M - T*E
      IF(E.NE.0) CALL READJT(B,E,K)
      RETURN
      END
      SUBROUTINE RECIP(A,BP,K)
      INTEGER A(1), BP(1)
C        THE PROCEDURE IS THE LONG EQUIVALENT OF B = 1./A.  THE ORDINARY
C     DIVISION OPERATION IS FIRST USED UPON THE SHORTENED REPRESENTATION
C     OF A TO GET A FIRST APPROXIMATION.  SUCCESSIVE APPROXIMATIONS ARE
C     THEN GENERATED BY THE ITERATIVE PROCESS B = B*(2.0 - B*A).  IF A=0
C     THE ORDINARY FAILURE FOR A DIVISION BY ZERO OF THE SYSTEM IN USE
C     WILL OCCUR.
      COMMON/LONG/N,W,T,S,R,TR,EX,RM
      INTEGER   N,W,T,S,EX
      REAL R, TR,RM
      INTEGER C(50),D(50),E(50),I,J,JJ,B(50)
C        THE DIMENSIONS HERE LIMIT THE PROCEDURE TO K NOT GREATER THAN
C     50.
      I = A(2)
      A(2) = 0
      CALL SHORTN(A,X,&2,K)
      X=1.0/X
      CALL LNGTHN(X,B,K)
    2 A(2) = I
      A(2) = I
      B(2) = B(2) - I
      D(1) = 1
      D(2) = 0
      D(3) = 2
      CALL COPY(B,E,K)
      DO 1 J = 4,K
    1 D(J) = 0
      GO TO 5
C     5 IS ALGOL HH. 4 IS ALGOL GG.
    4 CALL COPY(B,E,K)
      CALL COPY(C,B,K)
    5 CALL MLTPLY(A,B,C,K)
      CALL SBTRCT(D,C,C,K)
      CALL MLTPLY(B,C,C,K)
      DO 10 JJ = 1,K
      J = K + 1 - JJ
      IF(C(J).NE.B(J)) GO TO 15
   10 CONTINUE
      GO TO 20
   15 DO 11 JJ = 1,K
      J = K + 1 - JJ
      IF(C(J).NE.E(J)) GO TO 4
   11 CONTINUE
```

```
   20 CALL COPY(B,BP,K)
      RETURN
      END
      SUBROUTINE DIVIDE(A,B,C,K)
      INTEGER A(1),B(1),C(1)
C        THE PROCEDURE IS THE LONG EQUIVALENT OF C = A/B. IF B = 0, THE
C     ORDINARY FAILURE FOR DIVISION BY ZERO OF THE SYSTEM IN USE WILL
C     OCCUR.
      COMMON/LONG/N,W,T,S,R,TR,EX,RM
      INTEGER   N,W,T,S,EX
      REAL R, TR,RM
      INTEGER CL(50)
C        THE DIMENSIONS HERE LIMIT THE PROCEDURE TO K NOT GREATER THAN
C     50
      CALL RECIP(B,CL,K)
      CALL MLTPLY(A,CL,C,K)
      RETURN
      END
```

ENTRY NAME
LSQRT

IDENTIFICATION *** Multiple Precision Square Root

PURPOSE *** Given a multiple precision number X of arbitrary array length K(see description of Multiple Precision Basic Arithmetic Package for storage details), LSQRT uses Newton's method for finding the root of an equation to evaluate the square root of X.

OTHER SUBROUTINES USED *** SQRT, IBCOM, COPY, SHORTN, LNGTHN, MLTINT, RECIP, MLTPLY, SBTRCT and ADD. An internal subroutine OUTPUT is also called.

USAGE *** INTEGER X(KX),Y(KX),K

```
        .
        .
        .
        CALL LNGCNS(K,4,8,75)
        .
        .
        .
        CALL LSQRT(X,Y,K)
```

where the parameters in the calling sequence are:

X - The multiple precision argument. X.GE.0

input

Y - On output Y will contain the multiple precision square root of X. The ordinary system error exit will occur if X is negative.

output

K - The array length of the multiple precision numbers X and Y. K.LE.KX.LE.50

input

ERROR MESSAGES *** If Newton's method does not converge in 50 steps the following message is printed and control returns to the calling program.

```
NO CONVERGENCE FOR SQUARE ROOT
X = ...
```

ACCURACY *** The maximum relative error in tests run was .32*10**(-(K-3)*4).

EXECUTION TIME *** Using the FORTRAN G compiler the execution time in seconds is given approximately by .0017*K**2.

```
      SUBROUTINE LSQRT(X,Y,K)
      INTEGER X(1),Y(1)
C        THIS SUBROUTINE IS THE LONG ARITHMETIC EQUIVALENT OF Y=SQRT(X).
C     THE NORMAL SINGLE PRECISION ERROR EXIT FOR THE SQUARE ROOT OCCURS
C     IF X IS NEGATIVE.  THE ROUTINE USES THE QUADRATICALLY CONVERGENT
C     ITERATION Y(J+1) = Y(J) + Y(J)*(X - Y(J)**2)/(2*X), WHICH MINIM-
C     IZES DIVISIONS.  THE STARTING VALUE IS OBTAINED FROM THE SINGLE
C     PRECISION SQUARE ROOT ROUTINE.
      COMMON/LONG/N,W,T,S,R,TR,EX,RM
      INTEGER N,W,T,S,EX
      REAL R,TR,RM
      INTEGER R2X(50),B(50),DELTA(50),LI
      INTEGER OLDDEL(50)
C          THIS SET OF DIMENSIONS LIMITS K TO 50 OR LESS.
      IF(X(1)) 100,101,100
  101 CALL COPY(X,Y,K)
      RETURN
  100 CALL COPY(X,DELTA,K)
      IF (DELTA(2).GE.0)GO TO 1
      LI = DELTA(2) / 2.0
      LI = DELTA(2)/2 + DELTA(2) - 2*LI
      GO TO 4
1     LI = DELTA(2)/2
4     DELTA(2) = DELTA(2) - 2*LI
C     DELTA NOW CONTAINS X WITH EXPONENT 0 OR 1. LI IS THE EXPONENT FOR
C     THE SQUARE ROOT.
      CALL SHORTN(DELTA,XX,&2,K)
      XX=SQRT(XX)
      CALL LNGTHN(XX,B,K)
2     CALL MLTINT(X,2,R2X,&3,K)
    3 CALL RECIP(R2X,R2X,K)
      B(2) = LI
      ISW=0
      DO 10 J=1,50
      CALL MLTPLY(B,B,DELTA,K)
      CALL SBTRCT(X,DELTA,DELTA,K)
      CALL MLTPLY(B,DELTA,DELTA,K)
      CALL MLTPLY(R2X,DELTA,DELTA,K)
      IF(DELTA(1).EQ.0) GO TO 15
      CALL ADD(B,DELTA,B,K)
      IF(B(2) - DELTA(2) - K + 3) 10,5,15
    5 IF (ISW.NE.0)GO TO 6
      ISW = 1
      GO TO 7
    6 CALL ADD(DELTA,OLDDEL,OLDDEL,K)
      IF (OLDDEL(1).EQ.0)GO TO 20
    7 CALL COPY(DELTA,OLDDEL,K)
   10 CONTINUE
      WRITE(6,999)
  999 FORMAT(10X,30HNO CONVERGENCE FOR SQUARE ROOT     )
      CALL OUTPUT(X,6,K)
      WRITE(6,998)
  998 FORMAT(6H+  X =)
   15 CALL COPY(B,Y,K)
      RETURN
   20 OLDDEL(1) = 1
      OLDDEL(2) = -1
```

```
      OLDDEL(3) = 5000
      DO 30 I=4,K
   30 OLDDEL(K) = 0
      CALL MLTPLY(DELTA,OLDDEL,DELTA,K)
      CALL SBTRCT(B,DELTA,Y,K)
      RETURN
      END
```

ENTRY NAME
LNGLOG

IDENTIFICATION *** Multiple Precision Natural Logarithm

PURPOSE *** Given a multiple precision number X of arbitrary array length K(see description of Multiple Precision Basic Arithmetic Package for storage details), LNGLOG uses an infinite series to evaluate the natural logarithm of X.

OTHER SUBROUTINES USED *** FIXPI, IBCOM, MLTINT, COPY, ADD, SET, DIVIDE, SBTRCT and MLTPLY. An internal subroutine OUTPUT is also called.

USAGE ***
```
INTEGER X(KX),Y(KX),K
  :
CALL LNGCNS(K,4,8,75)
  :
CALL LNGLOG(X,Y,&N,K)
```

where the parameters in the calling sequence are:

X - The multiple precision argument. X.GT.0 input

Y - On output Y will contain the multiple precision natural logarithm of X. output

N - Control returns to the statement labeled N(an actual integer) if X.LE.0. input

K - The array length of the multiple precision numbers X and Y.

K.LE.KX.LE.50

input

ERROR MESSAGE *** If the series does not converge in 1500 terms control returns to the calling program after the following message has been printed:

```
NO CONVERGENCE FOR LOGARITHM
X =
```

ACCURACY *** The maximum relative error observed in tests run was .30*10**(-((K-3)*4-1)).

EXECUTION TIME *** Using the FORTRAN G compiler the execution time in seconds is given approximately by .0026*K**3.

```
      SUBROUTINE LNGLOG(X,Y,*,K)
C        THIS SUBROUTINE IS THE LONG EQUIVALENT OF Y = ALOG(X).  IF X IS
C     NONPOSITIVE, IT RETURNS TO THE LABEL PARAMETER.
C     FOR.3.LE.X.LE.3, THE SERIES LN(X ) = 2*SUM(K=0,NU-1)((1/(2K+1)*
C     RATIO**(2K+1)) + REM IS USED, WHERE RATIO = (X-1)/(X+1), AND REM =
C     2/(2NU+1)*RATIO**(2NU+1)/(1-RATIO**2).  FOR OTHER X, THE RANGE IS
C     REDUCED USING VALUES OF LOG10 AND LOGT COMPUTED ON THE FIRST ENTRY
      COMMON/LONG/N,W,T,S,R,TR,EX,RM
      REAL R,TR,RM
      INTEGER N,W,T,S,EX
      INTEGER X(1),Y(1)
      INTEGER SUM(50),LOG10(50),LOGT(50),TERM(50),RATSQR(50),RATIO(50)
C     THESE DIMENSIONS LIMIT K TO 50.
      INTEGER X2,J,ED,SW,XEXP
      DATA LOG10/1,0,2,3025,8509,2994,0456,8401,7991,4546,8436,4207,
     1             6011,0148,8628,7729,7603,3327,9009,6757,2609,6773,
     2             5248,0235,9972,0508,9598,2983,4196,7784,0422,8624,
     3             8633,4095,2546,5082,8067,5666,6287,3690,9878,1689,
     4             4829,0720,8325,5546,8084,3799,8948,2623/
      DATA LOGT /   1,   0,   9,2103,4037,1976,1827,3607,1965,8187,3745,
     1 6830,4044, 595,4515, 919, 413,3311,6038,7029, 438,7094, 992, 943,
     2 9888,2035,8393,1933,6787,1136,1691,4499,4533,6381, 186, 331,2270,
     3 2666,5149,4763,9512,6757,9316,2883,3302,2187,2337,5199,5793, 492/
C        HERE TO COMPUTE LOG(X).
      IF(X(1).LE.0) RETURN 1
      XEXP = X(2)
      X(2) = 0
      CALL MLTINT(LOGT,XEXP,SUM,&11,K)
   11 J = -1
      ED = X(3)
C        FIND NO OF POWERS OF 10 IN INTEGER PART OF X.
   12 ED = ED/10
      J = J + 1
      IF(ED) 13,13,12
   13 IF(X(3).GT.3*10**J) J = J + 1
      IF(J) 17,17,14
   17 CALL COPY(X,TERM,K)
      GO TO 21
   14 CALL MLTINT(LOG10,J,TERM,&15,K)
   15 CALL ADD(SUM,TERM,SUM,K )
      CALL SET(TERM,10**J,&16,K)
   16 CALL DIVIDE(X,TERM,TERM,K )
   21 X(2) = XEXP
C        IS TERM = 1
      IF(TERM(3) - 1) 20,18,20
   18 DO 19 J = 4,K
      IF(TERM(J))20,19,20
   19 CONTINUE
      GO TO 170
   20 CALL SET(RATSQR,1,&28,K )
   28 CALL SBTRCT(TERM,RATSQR,RATIO,K)
      CALL ADD(TERM,RATSQR,RATSQR,K)
      CALL DIVIDE(RATIO,RATSQR,RATIO,K)
   50 CALL MLTPLY(RATIO,RATIO,RATSQR,K)
      CALL MLTINT(RATIO,2,RATIO,&51,K)
   51 CALL ADD(RATIO,SUM,SUM,K)
C        THIS LOOP SUMS THE SERIES FOR THE SCALED RATIO.
```

```
      DO 60 J = 1,1500
      CALL MLTPLY(RATIO,RATSQR,RATIO,K)
C        RATIO WILL NOW BE 2*((X-1)/(X+1))**(2J+1)
      CALL SET(TERM,2*J+1,&52,K)
   52 CALL DIVIDE(RATIO,TERM,TERM,K)
      IF(TERM(1).EQ.0) GO TO  170
      IF(SUM(2 ) - TERM(2) -K + 2) 53,54,54
   53 CALL ADD(SUM,TERM,SUM,K)
   60 CONTINUE
      WRITE(6,999)
  999 FORMAT(30H NO CONVERGENCE FOR LOGARITHM    )
      CALL OUTPUT(X,6,K)
      WRITE(6,998)
  998 FORMAT(15H+    X =                      )
   54 CALL SET(RATIO,1,&55,K)
   55 CALL SBTRCT(RATIO,RATSQR,RATIO,K)
      IF(RATIO(1)) 56,57,56
   56 CALL DIVIDE(TERM,RATIO,TERM,K)
      CALL ADD(SUM,TERM,SUM,K)
   57 CONTINUE
  170 CALL COPY(SUM,Y,K)
      RETURN
      END
```

ENTRY NAME
OUTPUT

IDENTIFICATION *** This subroutine is used internally by LSQRT and LNGLOG to print error messages.

OTHER SUBROUTINES USED *** IBCOM

```
      SUBROUTINE OUTPUT(A,UNIT,K)
      INTEGER A(1),UNIT,IT,J
C        THIS SUBROUTINE OUTPUTS THE LONG NUMBER, A, INTHE FOLLOWING
C     FORMAT.. 15 SPACES FOR DESCRIPTION, (DECIMAL EXPONENT), INTEGER
C     PART WITH SIGN, DECIMAL POINT, FRACTIONAL PART IN 5-DIGIT BLOCKS,
C     SEPARATED BY SPACES, AND WITH INTERNAL SPACES SUPPRESSED.
      COMMON/LONG/N,W,T,S,R,TR,EX,RM
      INTEGER   N,W,T,S,EX
      REAL R, TR,RM
  999 FORMAT(15X,1H(,I6,1H),2X,I6,1H.,10(I5,1X),(/26X,11(I5,1X)))
      IT = A(1)*A(3)
      WRITE(UNIT,999)A(2),IT,(A(J), J = 4,K)
      RETURN
      END
```

ENTRY NAME
EXPO

IDENTIFICATION *** Multiple Precision Exponentiation with Integer Exponent

PURPOSE *** Given a multiple precision number X of arbitrary array length K(see description of Multiple Precision Basic Arithmetic Package for storage details) and an ordinary integer I, EXPO will evaluate X**I by successive multiplications.

OTHER SUBROUTINES USED *** IBCOM, SET, SBTRCT, COPY, MLTPLY and RECIP.

USAGE *** INTEGER X(KX),Y(KX),I,K

```
      .
      .
      .
      CALL LNGCNS(K,4,8,75)
      .
      .
      .
      CALL EXPO(X,I,Y,K)
```

where the parameters in the calling sequence are:

X - The multiple precision argument.

input

I - The integer exponent.

input

Y - On output Y=X**I where Y is a multiple precision number.

output

K - The array length of the multiple precision numbers X and Y.
K.LE.KX.LE.50

input

ERROR MESSAGE *** If X.EQ.I.EQ.0 control returns immediately to the calling program after the following message has been printed:

*** ERROR IN EXP0 (0**0) ***

ACCURACY *** The maximum relative error observed in tests run was .32*10**(-((K-3)*4-1)).

EXECUTION TIME *** Using the FORTRAN G compiler the execution time in seconds is given approximately by .69*10**(-4)*K**2*I. This is a good approximation for small I, but when I is very large it over estimates the time required. Note that EXP0 requires much less execution time than LEXP0 and should be used whenever possible.

```
      SUBROUTINE EXPO(XX,A,Y,K)
C
C *** Y = XX ** A WHERE A IS AN INTEGER , K.LE.50
C
      COMMON /LONG/LN,LW,LT,LS,LR,LTR,LEX,LRM
      INTEGER LN,LW,LT,LS,LEX
      REAL LR,LTR,LRM
      INTEGER XX(1),Z,Y(1),K,SW,N,A ,X(50)
      CALL SET(X,1,&5,K)
    5 CALL SBTRCT(XX,X,X,K)
      IF (X(1).EQ.0)GO TO 40
      CALL COPY(XX,X,K)
      IF (XX(1).EQ.0)GO TO 80
      SW = 0
      CALL SET(Y,1,&10,K)
C
C *** X**0 = 1
10    IF(A.EQ.0)RETURN
      IF (A.GT.0)GO TO 20
      SW = 1
      A = -1 * A
20    DO 30 N=1,A
      CALL MLTPLY(Y,X,Y,K)
30    CONTINUE
      IF (SW.EQ.0)RETURN
      CALL RECIP(Y,Y,K)
      A = -1 * A
      RETURN
C
C *** 1**A = 1
   40 CALL SET(Y,1,&50,K)
   50 RETURN
C
C *** 0**0 IS AN ERROR
   70 WRITE (6,1000)
 1000 FORMAT ('- *** ERROR IN EXPO (0**0) *** ')
      RETURN
   80 IF (A.EQ.0)GO TO 70
C
C *** 0**A = 0
      CALL SET(Y,0,&90,K)
   90 RETURN
      END
```

ENTRY NAME

LEXPO

IDENTIFICATION *** Multiple Precision Exponentiation

PURPOSE *** Given two multiple precision numbers X and Y of arbitrary array length K(see description of Multiple Precision Basic Arithmetic Package for storage details), LEXPO evaluates X**Y by the equation exp(Y * ln X) using the library subroutines EXP and LNGLOG.

USAGE *** INTEGER X(KX),Y(KX),Z(KX),K

```
          .
          .
          .
          CALL LNGCNS(K,4,8,75)
          .
          .
          .
          CALL LEXPO(X,Y,Z,K)
```

where the parameters in the calling sequence are:

X - The multiple precision argument.

input

Y - The multiple precision exponent.

input

Z - On output Z=X**Y where Z is a multiple precision number.

output

K - The array length of the multiple precision numbers X, Y and Z. K.LE.KX.LE.50

input

OTHER SUBROUTINES USED *** IBCOM, SET, SBTRCT, SHORTN, EXPO, LNGLOG, MLTPLY and LEXP.

ERROR MESSAGES *** Control is returned immediately to the calling program if either of the following messages is printed:

*** ERROR IN LEXPO (0**0) ***

*** ERROR IN LEXPO (X**A WHERE X IS NEGATIVE AND A IS NOT AN INTEGER) ***

ACCURACY *** The maximum relative error observed in tests run was .24*10**(-((K-3)*4-2)).

EXECUTION TIME *** Using a FORTRAN G compiler the execution time in seconds is given approximately by .0032*K**3.

```
      SUBROUTINE LEXPO(X,A,Y,K)
C
C *** Y = X ** A , K.LE.50
C
      COMMON /LONG/LN,LW,LT,LS,LR,LTR,LEX,LRM
      INTEGER LN,LW,LT,LS,LEX
      REAL LR,LTR,LRM
      INTEGER X(1),A(1),Y(1),T(50),K,AA
      REAL AS
      IF (A(1).EQ.0)GO TO 20
      IF (X(1).EQ.0)GO TO 50
      IF (A(2).LT.0.OR.A(2).GT.2.OR.A(2).EQ.2.AND.A(3).GT.20)GO TO 6
      CALL SET(T,1,&4,K)
    4 CALL SBTRCT(X,T,T,K)
      IF (T(1).EQ.0)GO TO 70
      CALL SHORTN(A,AS,&5,K)
    5 AA = AINT(AS)
      IF (AS.NE.AA)GO TO 6
C
C *** SPECIAL CASE WHERE A IS AN INTEGER
      CALL EXPO(X,AA,Y,K)
      RETURN
    6 IF (X(1).GT.0)GO TO 9
C
C *** ERROR RETURN WHEN X IS NEGATIVE AND A IS NOT AN INTEGER
      WRITE (6,1001)
 1001 FORMAT ('- *** ERROR IN LEXPO (X**A WHERE X IS NEGATIVE AND A IS N
     1OT AN INTEGER) ***')
      RETURN
C
C *** X**A = EXP(A * LN(X))
    9 CALL LNGLOG(X,T,&10,K)
   10 CALL MLTPLY(T,A,Y,K)
      CALL LEXP(Y,Y,K)
      RETURN
   20 IF (X(1).EQ.0)GO TO 40
C
C *** X**0 = 1
      CALL SET(Y,1,&30,K)
   30 RETURN
C
C *** 0**0 IS AN ERROR
   40 WRITE (6,1000)
 1000 FORMAT ('- *** ERROR IN LEXPO ( 0**0 ) ***')
      RETURN
C
C *** 0**A = 0
   50 CALL SET(Y,0,&60,K)
   60 RETURN
C
C *** 1**A = 1
   70 CALL SET(Y,1,&80,K)
   80 RETURN
      END
```

ENTRY NAME
LEXP

IDENTIFICATION *** Multiple Precision Exponential Function

PURPOSE *** Given a multiple precision number X of arbitrary array length K(see description of Multiple Precision Basic Arithmetic Package for storage details), LEXP uses a continued fraction to evaluate e**X.

OTHER SUBROUTINES USED *** IBCOM, COPY, SET, SHORTN, SBTRCT, MLTPLY, MLTINT, DIVIDE, ADD, RECIP and EXPO.

USAGE *** INTEGER X(KX),Y(KX),K

```
      .
      .
      .
      CALL LNGCNS(K,4,8,75)
      .
      .
      .
      CALL LEXP(X,Y,K)
```

where the parameters in the calling sequence are:

X - The multiple precision argument. ABS(X).LE.20*10**8

input

Y - On output Y=e**X where Y is a multiple precision number.

output

K - The array length of the multiple precision numbers X and Y. K.LE.KX.LE.50

input

ERROR MESSAGES *** If ABS(X).GT.20*10**8 control is returned immediately to the calling program after the following message has been printed:

*** X IS OUT OF RANGE WHEN CALCULATING E**X IN LEXP ***

If the continued fraction does not converge in 1000 steps control is returned to the calling program after the following message has been printed:

*** NO CONVERGENCE FOR EXP ***

ACCURACY *** The maximum relative error observed in tests run was .64*10**(-((K-3)*4-1)).

EXECUTION TIME *** Using the FORTRAN G compiler the execution time in seconds is given approximately by .0020*K**3.

```
      SUBROUTINE LEXP(X,W,K)
C
C *** W = E ** X FOR ABS(X).LE.20*10**8 , K.LE.50
C
      COMMON /LONG/LN,LW,LT,LS,LR,LTR,LEX,LRM
      INTEGER LN,LW,LT,LS,LEX
      REAL LR,LTR,LRM
      INTEGER X(1),W(1),U(50),V(50),SW,A(50),B1(50),B2(50)
      INTEGER ONE(50),AA(50),SW2,E1(50),HALF(50),SW3
      REAL*4 XS
      DATA E1/1,0,2,7182,8182,8459,0452,3536,0287,4713,5266,2497,7572,
     1 4709,3699,9595,7496,6967,6277,2407,6630,3535,4759,4571,3821,7852,
     2 5166,4274,2746,6391,9320,0305,9921,8174,1359,6629,0435,7290,0334,
     3 2952,6059,5630,7381,3232,8627,9434,9076,3233,8298,8075/
C
C *** CHECK IF X IS IN CORRECT RANGE
      IF (X(2).GT.2.OR.X(2).EQ.2.AND.X(3).GT.20)GO TO 160
C
C *** E ** X IS CALCULATED FROM A CONTINUED FRACTION FOR 0.LT.X.LE.0.5
C     FOR OTHER VALUES, X IS REDUCED USING A PRECOMPUTED VALUE OF E ** 1
      CALL COPY(X,AA,K)
      CALL COPY(X,A,K)
      CALL SET(ONE,1,&5,K)
      HALF(1) = 1
      HALF(2) = -1
      HALF(3) = 5000
      DO 4 I=4,K
    4 HALF(I) = 0
      SW3 = 0
5     CALL SHORTN(X,XS,&10,K)
   10 IF (XS.NE.0.0)GO TO 11
      CALL COPY(ONE,W,K)
      RETURN
   11 IF (XS.GE.1.0.OR.XS.LT.0.0)GO TO 14
      SW2 = 2
   12 CALL SBTRCT(A,HALF,A,K)
      IF (A(1).GT.0)GO TO 13
      CALL COPY(AA,A,K)
      GO TO 20
   13 CALL COPY(AA,A,K)
      CALL MLTPLY(A,HALF,A,K)
      SW3 = 1
      GO TO 20
   14 IF (XS.NE.1)GO TO 110
      CALL COPY(E1,W,K)
      RETURN
C
C *** U(1) = V(1) = W(1) = 1
   20 CALL COPY(ONE,U,K)
      CALL COPY(ONE,V,K)
      CALL COPY(ONE,W,K)
      SW = -1
      CALL COPY(ONE,B1,K)
      DO 70 I=1,1000
C
C *** U(I+1)=1/(1+(A(I+1)/(B(I)*B(I+1)))*U(I))
      IF (SW.GT.0)GO TO 40
```

```
      CALL SET(B2,I,&30,K)
   30 GO TO 50
   40 CALL SET(B2,2,&50,K)
   50 CALL MLTINT(A,-1,A,&60,K)
   60 CALL MLTPLY(A,U,U,K)
      CALL MLTPLY(B1,B2,B1,K)
      CALL DIVIDE(U,B1,U,K)
      CALL ADD(U,ONE,U,K)
      CALL RECIP(U,U,K)
C
C *** V(I+1) = V(I)*(U(I+1)-1)
      CALL SBTRCT(U,ONE,B1,K)
      CALL MLTPLY(V,B1,V,K)
C
C *** W(I+1) = W(I)+V(I+1)
      CALL ADD(W,V,W,K)
C
C *** CHECK TO SEE IF FINISHED
      IF (V(2).LE.(W(2)-(K-2)))GO TO 120
      CALL COPY(B2,B1,K)
      SW=-SW
   70 CONTINUE
      WRITE (6,1000)
 1000 FORMAT ('-*** NO CONVERGENCE FOR EXP ***')
      RETURN
  110 SW2 = 1
      INT = AINT(XS)
      CALL SET(A,INT,&120,K)
      CALL SBTRCT(AA,A,A,K)
      IF (A(1).NE.0)GO TO 150
      CALL COPY(ONE,W,K)
  120 IF (SW3.NE.1)GO TO 130
      CALL MLTPLY(W,W,W,K)
  130 IF (SW2.EQ.2)RETURN
      CALL EXPO(E1,INT,A,K)
      CALL MLTPLY(A,W,W,K)
      RETURN
  140 CALL COPY(E1,W,K)
      RETURN
  150 CALL COPY(A,AA,K)
      GO TO 12
  160 WRITE (6,1001)
 1001 FORMAT ('-*** X IS OUT OF RANGE WHEN CALCULATING E**X IN LEXP ***'
     1)
      RETURN
      END
```

ENTRY NAME
LCOS

IDENTIFICATION *** Multiple Precision Cosine

PURPOSE *** Given a multiple precision number X of arbitrary array length K(see description of Multiple Precision Basic Arithmetic Package for storage details), LCOS uses an infinite series to evaluate the cosine of X.

OTHER SUBROUTINES USED *** IBCOM, COPY, SET, SBTRCT, MLTPLY, MLTINT, DIVIDE and ADD.

USAGE *** INTEGER X(KX),Y(KX),K

```
      .
      .
      .
      CALL LNGCNS(K,4,8,75)
      .
      .
      .
      CALL LCOS(X,Y,K)
```

where the parameters in the calling sequence are:

X - The multiple precision argument. input

Y - On output Y=cos(X) where Y is a multiple precision number. output

K - The array length of the multiple precision numbers X and Y. input
K.LE.KX.LE.50

ERROR MESSAGES *** If the series for cosine does not converge in 1000 steps, control is returned to the calling program after the following message has been printed:

*** COSINE DOES NOT CONVERGE ***

ACCURACY *** The maximum relative error observed in tests run was .20*10**(-((K-3)*4-1)).

EXECUTION TIME *** Using the FORTRAN G compiler the execution time is given approximately by .00068*K**3 seconds.

```
      SUBROUTINE LCOS(XA,Y,K)
C
C *** Y = COS(XA) , K.LE.50
C
      INTEGER       Y(1),K,NUM(50),DENOM(50),TERM(50),X2(50),FACT,FACT2,
     1         SW,ONE(50),XA(1),X(50),TEMP(50),SIGN
      INTEGER PI2(50),TERM2(50,10)          ,PIC(50)
      DATA   PIC/   1,   0,   6,2831,8530,7179,5864,7692,5286,7665,5900,
     1 5768,3943,3879,8750,2116,4194,9889,1846,1563,2812,5724,1799,7256,
     2  696,5068,4234,1359,6429,6173, 265,6461,3294,1876,8921,9101,1644,
     3 6345, 718,8162,5696,2234,9005,6820,5403,8770,4221,1119,2892,4588/
      CALL COPY(XA,X,K)
      CALL SET(ONE,1,&10,K)
   10 IF (X(1).NE.0)GO TO 30
C
C *** COS(0) = 1
      CALL COPY(ONE,Y,K)
      RETURN
C
C *** REDUCE XA TO 0.LE.XA.LT.2*PI
   30 KK = 0
   31 CALL SBTRCT(X,PIC,X2,K)
      IF (X2(1).LT.0)GO TO 50
      CALL COPY(X2,X,K)
      GO TO 31
C
C *** THIS ENTRY TO SERIES FOR SIN(X), 0.LE.X.LE.PI/4
   36 CALL MLTPLY(X,X,X2,K)
      CALL COPY(X,Y,K)
      CALL COPY(X,NUM,K)
      CALL COPY(ONE,DENOM,K)
      SW = -1
      FACT = 2
      GO TO 33
C
C *** THIS ENTRY TO SERIES FOR COS(X), 0.LE.X.LE.PI/4
   32 CALL MLTPLY(X,X,X2,K)
      CALL COPY(ONE,Y,K)
      CALL COPY(ONE,NUM,K)
      CALL COPY(ONE,DENOM,K)
      SW = -1
      FACT = 1
C
C *** EVALUATE INFINITE SERIES FOR COS(X) OR SIN(X)
   33 DO 40 I=1,1000
      CALL MLTPLY(NUM,X2,NUM,K)
      FACT2 = FACT * (FACT + 1)
      IF (FACT2.LT.10000)GO TO 340
      CALL SET(TEMP,FACT2,&41,K)
      CALL MLTPLY(DENOM,TEMP,DENOM,K)
      GO TO 34
  340 CALL MLTINT(DENOM,FACT2,DENOM,&34,K)
   34 CALL DIVIDE(NUM,DENOM,TERM,K)
      CALL MLTINT(TERM,SW,TERM,&35,K)
   35 IF(KK.GT.0)GO TO 37
      CALL ADD(Y,TERM,Y,K)
      GO TO 39
```

```
   37 CALL COPY(TERM,TERM2(1,KK),K)
      IF (KK.LT.10)GO TO 39
      DO 38 II=1,9
      KK = 10-II+1
      CALL ADD(TERM2(1,KK),TERM2(1,KK-1),TERM2(1,KK-1),K)
   38 CONTINUE
      CALL ADD(TERM2(1,1),Y,Y,K)
      KK = 0
   39 IF (TERM(2).LE.-1*(K-2)+  Y(2))GO TO 170
      KK = KK + 1
      FACT = FACT + 2
      SW = -SW
   40 CONTINUE
   41 WRITE (6,1000)
 1000 FORMAT ('-  *** COSINE DOES NOT CONVERGE ***')
      RETURN
C
C *** REDUCE XA TO 0.LE.XA.LE.PI/4
   50 CALL SET(TEMP,8,&60,K)
   60 CALL DIVIDE(PIC,TEMP,PI2,K)
      DO 80 II=1,7
      I = 7 - II + 1
      CALL MLTINT(PI2,I,TEMP,&70,K)
   70 CALL SBTRCT(X,TEMP,TEMP,K)
      IF (TEMP(1).GT.0)GO TO 90
   80 CONTINUE
      SIGN = +1
      GO TO 32
   90 GO TO (100,110,120,130,140,150,160),II
  100 SIGN = +1
      TEMP(1) = -TEMP(1)
      CALL ADD(TEMP,PI2,X,K)
      GO TO 32
  110 SIGN = +1
      CALL COPY(TEMP,X,K)
      GO TO 36
  120 SIGN = -1
      TEMP(1) = -TEMP(1)
      CALL ADD(TEMP,PI2,X,K)
      GO TO 36
  130 SIGN = -1
      CALL COPY(TEMP,X,K)
      GO TO 32
  140 SIGN = -1
      TEMP(1) = -TEMP(1)
      CALL ADD(TEMP,PI2,X,K)
      GO TO 32
  150 SIGN = -1
      CALL COPY(TEMP,X,K)
      GO TO 36
  160 SIGN = +1
      TEMP(1) = -TEMP(1)
      CALL ADD(TEMP,PI2,X,K)
      GO TO 36
  170 IF (KK.EQ.0)GO TO 200
      IF (KK.EQ.1)GO TO 190
      KKM1 = KK-1
```

```
      DO 180 II=1,KKM1
      I = KK-II+1
      CALL ADD(TERM2(1,I),TERM2(1,I-1),TERM2(1,I-1),K)
180   CONTINUE
190   CALL ADD(TERM2(1,1),Y,Y,K)
200   Y(1) = SIGN
      RETURN
      END
```

ENTRY NAME
LGAM

IDENTIFICATION *** Multiple Precision Gamma Function

PURPOSE *** Given a multiple precision number X of arbitrary length K (see description of Multiple Precision Basic Arithmetic Package for storage details), LGAM uses the Sterling asymptotic expansion for natural log of gamma function and the library subroutine LEXP to evaluate the gamma function of X to D significant digits.

OTHER SUBROUTINES USED *** IBCOM, COPY, SET, SBTRCT, ADD, MLTPLY, LNGLOG, DIVIDE and LEXP.

USAGE *** INTEGER X(KX),Y(KX),D,K

```
      :
      CALL LNGCNS(K,4,8,75)
      :
      CALL LGAM(X,Y,D,K)
```

where the parameters in the calling sequence are:

X - The multiple precision argument. X.GT.0 input

Y - On output Y will contain the gamma function of X, where Y is a multiple precision number, to approximately D significant decimal digits (see ACCURACY). output

D - The number of significant decimal digits desired. For maximum efficiency set D = (K-3)*4+1. input

K - The array length of the multiple precision numbers X and Y. K.LE.KX.LE.50

input

ERROR MESSAGE *** If X is less than or equal to 0 control is returned immediately to the calling program after the following message is printed:

ERROR IN LGAM Z IS LESS THAN OR EQUAL TO 0
WHEN CALCULATING GAMMA (X)

ACCURACY *** The maximum relative error observed in tests run with D set equal to (K-3)*4+1 was .80*10**(-(K-4)*4). If D is set so as to be less than (K-4)*4 there should be D significant decimal digits in the answer.

EXECUTION TIME *** Using the FORTRAN G compiler the execution time in seconds is given approximately by .0059*K**3.

```
      SUBROUTINE LGAM(ZIN,Y,D,K)
C
C *** Y = GAMMA(ZIN) FOR ZIN.GT.0, K.LE.50, D(NO.SIGN.DIGITS).LE.(K-3)*4
C
      COMMON /LONG/LN,LW,LT,LS,LR,LTR,LEX,LRM
      INTEGER LN,LW,LT,LS,LEX
      REAL LR,LTR,LRM
      INTEGER ZIN(1),Y(1),Z(50),TEMP(50),TEMP2(50),TEMP3(50),C(50,71),
     1        LN2PI(50),ONE(50),D,K,SW,T(50)
C *** SET CONSTANTS C(I) = B(2I)/(2I*(2I-1)) WHERE B(2I) ARE THE
C       BERNOULLI NUMBERS
      INTEGER C1(250),C2(250),C3(250),C4(250),C5(250),C6(250),C7(250),
     1 C8(250),C9(250),C10(250),C11(250),C12(250),C13(250),C14(50)
      EQUIVALENCE (C(1,6),C1(1)),(C(1,11),C2(1)),(C(1,16),C3(1)),
     1(C(1,21),C4(1)),(C(1,26),C5(1)),(C(1,31),C6(1)),(C(1,36),C7(1)),
     2(C(1,41),C8(1)),(C(1,46),C9(1)),(C(1,51),C10(1)),(C(1,56),C11(1)),
     3(C(1,61),C12(1)),(C(1,66),C13(1)),(C(1,71),C14(1))
      DATA C/      1,  -1, 833,3333,3333,3333,3333,3333,3333,3333,3333,
     13333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,
     23333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,
     33333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,
     4            -1,  -1,  27,7777,7777,7777,7777,7777,7777,7777,7777,
     57777,7777,7777,7777,7777,7777,7777,7777,7777,7777,7777,7777,7777,
     67777,7777,7777,7777,7777,7777,7777,7777,7777,7777,7777,7777,7777,
     77777,7777,7777,7777,7777,7777,7777,7777,7777,7777,7777,7777,7778,
     8            1,  -1,   7,9365, 793,6507,9365, 793,6507,9365, 793,
     96507,9365, 793,6507,9365, 793,6507,9365, 793,6507,9365, 793,6507,
     A9365, 793,6507,9365, 793,6507,9365, 793,6507,9365, 793,6507,9365,
     B 793,6507,9365, 793,6507,9365, 793,6507,9365, 793,6507,9365, 794,
     C            -1,  -1,   5,9523,8095,2380,9523,8095,2380,9523,8095,
     D2380,9523,8095,2380,9523,8095,2380,9523,8095,2380,9523,8095,2380,
     E9523,8095,2380,9523,8095,2380,9523,8095,2380,9523,8095,2380,9523,
     F8095,2380,9523,8095,2380,9523,8095,2380,9523,8095,2380,9523,8095,
     G            1,  -1,   8,4175, 841,7508,4175, 841,7508,4175, 841,
     H7508,4175, 841,7508,4175, 841,7508,4175, 841,7508,4175, 841,7508,
     I4175, 841,7508,4175, 841,7508,4175, 841,7508,4175, 841,7508,4175,
     J 841,7508,4175, 841,7508,4175, 841,7508,4175, 841,7508,4175, 842/
      DATA C1/    -1,  -1,  19,1752,6917,5269,1752,6917,5269,1752,6917,
     15269,1752,6917,5269,1752,6917,5269,1752,6917,5269,1752,6917,5269,
     21752,6917,5269,1752,6917,5269,1752,6917,5269,1752,6917,5269,1752,
     36917,5269,1752,6917,5269,1752,6917,5269,1752,6917,5269,1752,6918,
     4            1,  -1,  64,1025,6410,2564,1025,6410,2564,1025,6410,
     52564,1025,6410,2564,1025,6410,2564,1025,6410,2564,1025,6410,2564,
     61025,6410,2564,1025,6410,2564,1025,6410,2564,1025,6410,2564,1025,
     76410,2564,1025,6410,2564,1025,6410,2564,1025,6410,2564,1025,6410,
     8            -1,  -1, 295,5065,3594,7712,4183,  65,3594,7712,4183,
     9  65,3594,7712,4183,  65,3594,7712,4183,  65,3594,7712,4183,  65,
     A3594,7712,4183,  65,3594,7712,4183,  65,3594,7712,4183,  65,3594,
     B7712,4183,  65,3594,7712,4183,  65,3594,7712,4183,  65,3594,7712,
     C            1,  -1,1796,4437,2368,8305,7316,4938,4900,1588,9396,
     D6943,5025,4721,7717,4963,5526,7253,1000,7043,7531,7378,4133,5364,
     F5551,7879,6664,8647,7631,9884,6790,1780,5952,7904,7291,4311,9235,
     F9984,9296,4437,2368,8305,7316,4938,4900,1588,9396,6943,5025,4722,
     G            -1,   0,   1,3924,3221,6905,9011,1642,7432,2169, 590,
     H1116,4274,3221,6905,9011,1642,7432,2169, 590,1116,4274,3221,6905,
     I9011,1642,7432,2169, 590,1116,4274,3221,6905,9011,1642,7432,2169,
     J 590,1116,4274,3221,6905,9011,1642,7432,2169, 590,1116,4274,3222/
```

```
 DATA C2/     1,   0,  13,4028,6404,4168,3919,9447,8951,   6,9013,
11124,9137,3360,9385,7832,9882,6777, 876,4665,2864, 441,6839,1994,
24789,5100, 690,1311,2491,3733,6093,8578,3298,8267,7708,7646,6528,
36404,4168,3919,9447,8951,   6,9013,1124,9137,3360,9385,7832,9883,
4            -1,   0, 156,8482,8462,6002, 173, 636,5132,4520,8897,
53828,1042,6288,6871,5825,2375,6436,7999,1506, 784,6260, 201,7306,
63651,3245,2088,9738,2810,4262,8868,7158,2523,7564,3679,9915, 607,
78462,6002, 173, 636,5132,4520,8897,3828,1042,6288,6871,5825,2376,
8            1,   0,2193,1033,3333,3333,3333,3333,3333,3333,3333,
93333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,
A3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,
B3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,3333,
C            -1,   1,   3,6108,7712,5372,4989,3571,7326,5219,2422,
D3073,6483,6100,4682,8437,6330,3533,4184,7594,7211,5793,9548,7441,
E4644,5295,8705,8322,6905, 659,8552,5755,6406,9816,9433,8016,1770,
F9663,6866,7518, 928, 544,9127,2882, 774,7977,8629,2039,1656, 238,
G            1,   1,  69,1472,2688,5131,3067,1083,9525, 775,6734,
H6755,3334, 716,8779,8050,4231,8946,6571,  16, 993,3756,7635,6766,
I4568,7769,1582,9196,1406,5331,5291,8024,9981,1398,5878,1683, 308,
J4877, 559,5603,2596,7444,9885,7958,2572,9084,9059,4808,7260,6568/
 DATA C3/    -1,   1,1523,8221,5394, 741,6192,2833,6495,8886,7805,
11865,9076,5338,3934,2188,4882,9854,5224,5414,2947,5015,8127,7672,
23592,6628,7160, 253,  44,2757,7482,6059,4560,4048, 708,4123,9721,
36951,2966,4769,1334,5983,5547,1220,7463,6306,1353,5736,8753,9531,
4            1,   2,   3,8290, 751,3914,1414,1414,1414,1414,1414,
51414,1414,1414,1414,1414,1414,1414,1414,1414,1414,1414,1414,1414,
61414,1414,1414,1414,1414,1414,1414,1414,1414,1414,1414,1414,1414,
71414,1414,1414,1414,1414,1414,1414,1414,1414,1414,1414,1414,1414,
8           -1,   2, 108,8226,6035,7843,9108,9015,1491,6552,5105,
93747,2943,4879,8108,1966, 443,7205,9409,6533,9461,5800,6308,3822,
A4823,1872,2920,2266,7962,5175,8657,7736,9536,7680,3300,8634,5123,
B9391, 890,1514,9165,5251, 537,4729,4348,7981, 819,6604,4372, 594,
C            1,   2,3473,2028,3765,  22,5225,2252,2522,5225,2252,
D2522,5225,2252,2522,5225,2252,2522,5225,2252,2522,5225,2252,2522,
E5225,2252,2522,5225,2252,2522,5225,2252,2522,5225,2252,2522,5225,
F2252,2522,5225,2252,2522,5225,2252,2522,5225,2252,2522,5225,2252,
G           -1,   3,  12,3696, 214,2269,2744,5425,1710,3492,7132,
H4881, 809,7864,1954,2517,1034,9271,3248,8108, 978,6419,5425,1710,
I3492,7132,4881, 809,7864,1954,2517,1034,9271,3248,8108, 978,6419,
J5425,1710,3492,7132,4881, 809,7864,1954,2517,1034,9271,3248,8108/
 DATA C4/    1,   3, 488,7880,6479,3079,3350,7581,5162,5180,2290,
12108,4705,3890,5673,8218, 703,6295,3273,5763,9974,1216,2066,5660,
28568,9333,4002,1582,4654,6869,8350,9607,2840,1778,5597,8844,5535,
31435,3368,4980,8934,7291,1947,9139,7368,1739,6650,4733,8655,6362,
4           -1,   4,   2,1320,3339,6091,9373,8969,7505,8982,1368,
53855,7465,4533,1985,1702, 559,4876,9801,1459,3865,8577,6879,  23,
65928,5473,5422,9861,8132,7940,6808,2237,9507,9204,5837,5463,4310,
77516,  94,3714,1894,1691,9447,2531,1762,7232,8951,8031,6818,3350,
8            1,   4, 102,1775,2965,2570,  77,5652,8762,8053,5855,
9  39,4011, 323, 890,4649,3301,8124,5074,8620,9613,8691,8833,7273,
A4436,5642,2379,8266,3514,5784, 819,5429,4720,2521,6706, 677,6989,
B7557,1315,9968,4791,1741,5287,6280,5358,5500,3940,1103,2308,9046,
C           -1,   4,5357,5472,1733,  20,3610,8277, 919,1969,2044,
D8484,9040,5436,5881,6499,8678,1401, 492,3584,2727,7075,5874,6357,
E2444,7989,7796,5436, 380,2677,6449,8052,3578,7713,8853,3591,2870,
F3308,9285,9053, 593, 337,2194,4860,8685,1543,6725,6995,5423,5224,
G            1,   5,  30,6157,8263,7048,8341,5043,1510,5132,9622,
```

```
H7581,9418,6765,6153,3704,3908,4724,7990,1051,3296,2275,8194,1867,
I6561,5337, 439, 847,2479,9010,5132,9622,7581,9418,6765,6153,3704,
J3908,4724,7990,1051,3296,2275,8194,1867,6561,5337, 439, 847,2480/
 DATA C5/    -1,   5,1899,9917,4263,9920,4050,2937,1429,3069,4290,
12947,3424,5899,6177, 871,8707,6088,2969,5400,1726,4767,5422,3702,
2 594,4012,8252,5588,8518,9295,8441,2381,3047,2314,7120,4834,1349,
31182,6365,7664,3235,9107,1648,7853,  28,3635,4667,6532,2481,1937,
4             1,   6,  12,7633,7403,3828,8341,4923,4951,3776,9782,
55976,5416,3360,8829,9014,4823,9746,8163,7707,1259,5605, 153,3358,
62033,3503,2199,6465,6575, 504,8435,2040,1282,7316,3691,8992, 993,
79593,9184, 888, 342, 150,6561,6522, 878,8918,4876,9372,7680,4870,
8            -1,   6, 925,2847,1761,2041,6307,2302,4234,8347,6227,
97951,9331,2434,6917,4503,6572,6227,7951,9331,2434,6917,4503,6572,
A6227,7951,9331,2434,6917,4503,6572,6227,7951,9331,2434,6917,4503,
B6572,6227,7951,9331,2434,6917,4503,6572,6227,7951,9331,2434,6917,
C             1,   7,   7,2188,2259,5185,6102,9783,6050,1873, 163,
D7922,4898,4042, 259,6887,6994,7467,5389, 375,6566,5576,3703, 429,
E1802,9537,1196,3524,6307,8600,4559,4211,5174,9430, 723,5603,1321,
F2409,5549,6084,8448,8056,2989,3943,8992,9626,3257, 125,8796,7093,
G            -1,   7, 604,5183,4059,9585,6967,7431,4823,8754,5472,
H8606,6144,3959,6719,6207,4063, 160,8096, 133,5195,3622,1003,4744,
I5201,3024,5555,7669,8196,9600,9766,7283,3550,6401, 980,9771,6333,
J9647,5422,4862,3194, 471, 212,1925,7736,5698,9317,2533,6909,5727/
 DATA C6/     1,   8,   5,4206,7047,1570, 945,4519,3477,8148,2610,
1  13,6612, 218,5792,3497,2677,5956,2841,5300,5464,4808,7431,6939,
28907,1038,2513,6612, 218,5792,3497,2677,5956,2841,5300,5464,4808,
37431,6939,8907,1038,2513,6612, 218,5792,3497,2677,5956,2841,5301,
4            -1,   8, 519,2957,8153,1408,1946,7001,9476,4391,8576,
58469,9706,2713,9744,7868, 361, 333, 220,9772,7980, 809,2125,7391,
68456,2713,9744,7868, 361, 333, 220,9772,7980, 809,2125,7391,8456,
72713,9744,7868, 361, 333, 220,9772,7980, 809,2125,7391,8456,2714,
8             1,   9,   5,3036,5885,5119,7005,9665,4839,2430,6975,
98643,6992,9263,5405,5490,9795,6625,5361,1937,1291,3375,4701,5678,
A 993,8997,4795,5555,8724,9292,6354, 554,9097,9566,2553,6119,3712,
B9133,7547, 156,7809,9389,9747,9555,5587,2492,9263,5405,5490,9796,
C            -1,   9, 576,3325,3481,6496,4013,8944,3585, 780,9925,
D5519, 737,5621,8905,4726,3681,5920,3980, 995, 248,7562,1890,5472,
E6368,1592, 398,  99,5024,8756,2189, 547,2636,8159,2039,8009,9502,
F4875,6218,9054,7263,6815,9203,9800,9950,2487,5621,8905,4726,3682,
G             1,  10,   6,6511,5571,4848,4539,3751,6520,1458,1055,
H5951, 397,3935,9454,9289,5890,9363, 734,7819,9014, 226,5149,9687,
I6295,1004,2247,3355,7250,3974,2186, 985,8536,3741,5180,5872,5652,
J1120,5750, 912,3692,5284,6901,3465,9516,1659,4604,9595,7740,2933/
 DATA C7/    -1,  10, 813,7378,3581,3668, 538,7161,7263,2093,5756,
19184, 689,1649,7387,9262,3679,4500,4499,7330,1502, 848,7282,1186,
28843,6915, 278,9566,2967,8117,9444,1362,7669,6163,7763,9570,8725,
3 851,2769,6905, 662,4003,7857,7760, 983,1819,8179, 794,   7,5166,
4             1,  11,  10,5369,6695,3357,1418, 375,4804,9276,4181,
5 189,6483,7337,5011,4155,2511,4155,2511,4155,2511,4155,2511,4155,
62511,4155,2511,4155,2511,4155,2511,4155,2511,4155,2511,4155,2511,
74155,2511,4155,2511,4155,2511,4155,2511,4155,2511,4155,2511,4155,
8            -1,  11,1441,8180,5999,6220,6261,8053,7780,1511,8128,
9 957, 332, 636,6421,1111,1111,1111,1111,1111,1111,1111,1111,1111,
A1111,1111,1111,1111,1111,1111,1111,1111,1111,1111,1111,1111,1111,
B1111,1111,1111,1111,1111,1111,1111,1111,1111,1111,1111,1111,1111,
C             1,  12,  20,8173,5652,2089,5654,6242,4808,2412,6356,
D2311,3173,4326,4149,9791,8933,5880,1134,5696,6983,1875,9801,5807,
```

```
7807,2640,6404,5257,5353,1178,4859,7052,9370,1677, 651,4903,6215,
9439,9171,2526,9747,3312,3015,2206,6440,6921,1881,1467,8168,9277,
          -1,  12,3167, 226,6348,8666,1827,4134,9556,7742,5613,
4291,8069,8304,2075,3030,3915,4461,6387, 695,7705,1960,8668,6379,
6235,6067, 724,8281,4451,1053,1793, 590,7998,4964,8804,4823,5877,
4714,8032,6391,9730,5125,4238,2582,1552, 624, 345, 813, 906,8015/
DATA C8/     1,  13,  50,7000,6461,2111,3734,3179,2648,1531,7487,
6567,6296,2804,4555,6213, 731,9401, 610,3426, 498,7852,6451,4849,
5215,4296,1971,3421,5875,8490,7531,3600, 793,2966,4336,3577,7678,
6156,9735,7330,5567,9508,1560,8111,4581,7839,2582,6763,8455, 548,
          -1,  13,8529,9728,2030, 551,8816,2084,  52,2162,2788,
8780,7044,7003,8282,4140, 895,4544,6374,8834,9557,3840,8707,1170,
8398,5768,4843,3095,5086,9312,1966,6997,2371,2423,3826,8191,7106,
 151,4785,6013,4377,1244,1310,7502,2978, 687,7334,7493,9119,6051,
           1,  14, 150,6417,2809,3405,9857,6695,1173,6037,9879,
 761, 193, 840,2505,4563,9808,4127,4509,8039,2156,8627,4509,8039,
2156,8627,4509,8039,2156,8627,4509,8039,2156,8627,4509,8039,2156,
8627,4509,8039,2156,8627,4509,8039,2156,8627,4509,8039,2156,8627,
          -1,  15,   2,7893,4947, 383,1636,8712,8838,1686,3127,
8171,2347,5688,8605,4688,1031,9303,8537,2100,9345,5145, 866,3271,
3605,7439,4450,7147,1754,7593,2445,7621,3765,7762,8788,6019,5370,
4795,5423, 375,4489,7963,  35,1696,8107,7064,4640,2267, 312,7837,
           1,  15, 540,9350,4352,8604,1500,5763,5618,7188,4152,
5823,3629, 276,9152,4237,8260,3718,8555,3658, 622,9009,9069,2524,
5197,6751,8605,7704,5140,4834,5718,8594,4427,4831,2380,7116,1002,
8105,3257,9136,7644,8233,2098,3638,4136,7599,4035, 104,5972,5746/
DATA C9/    -1,  16,  10,9753,3782,1508,5198,5501,6788,7261,7079,
5167, 999, 311,9478,2603,8689,6852,4499,9327,5855,3503,2343,5429,
8184, 854,1812,7971,3194,6068,1162,8088,2238,3290,4683,9685,1375,
5747,7983,  98,9790,3514,9247,9152, 536,2013,8726,5217, 524,5110,
           1,  16,2327,4876,2026,1847,9173,4786,4103,2052,1849,
3019,3480, 427,5712,3111,3439,9270,3512,7249,1039,4265,2329,7491,
 394,2652,3297,4910,3942,6523,2974,9103,9426,5232,9749,1039,4265,
2329,7491, 394,2652,3297,4910,3942,6523,2974,9103,9426,5232,9749,
          -1,  17,  51,5392,9162, 653,2139, 194,6552,1217,1717,
1770,3911,5939,6665,6336,7888,1319,6975,  52,9993,7743,9293,9556,
8347,4051,8631,8722,4331,1393, 353, 864,6045,6158,1996, 717,9850,
3894,5110,4397,9621,4268,7813,7484,2464,4980,7715,  83,1505,2999,
           1,  18,   1,1906,2102,3089, 226,4576,8481,6176,8713,
8046,2750,2271,7730,9854,7424,5633,6568,1311,7102,5008,5910,6529,
2096,2199,3127,1477,6632,3024, 549,8281,7869,4158, 756, 137,4570,
4467,3539,5189,  34,3642,6116,8384,8797,2508,5910,6529,2096,2199,
          -1,  18, 286,6893,8960,2966,7369,6226,4205,4190, 772,
4629,1381,9119,2049,3144,2655, 825,3986,3500,1121,9220,3375,1860,
 344,8829,7314,5799,4284,2769,1253,9738,8223,6708,5193,3678,2163,
 647,9132,7617,6102,4587,3072,1557,  41,8526,7011,5496,3981,2466/
DATA C10/    1,  19,   7,1893, 780,2337,2986,4392,8443,7139,3137,
1013, 925,7734,3048,7969,9797,2830,5521,9845,9798,4974,8451,7706,
2782,5611,9648, 258,2893,5182,  64,1145,9453,8510,7165,3628,6083,
5320,7026,6666, 563,4211,1326,2505,1686,8607,8039,1492,6860,2311,
          -1,  19,1876, 693,4305, 467,1847,8954,6272,6582,6766,
1208,6697,9299,2240,1628,4037,8099,2271,6173,4594,6573,8315, 225,
9266, 438,4197,3499,4199,1817,7932,4662,6366,2453,4408, 112,3526,
8974,7817, 605,1169,3228,3079,9291,6895,6463,3327,2272, 889, 517,
           1,  20,  50,9049,1469, 751, 704,8132,9151,8692,8962,
8896,1540,5955,7511,6088,2424,7587,1088,5216,6093,4162,4240,6469,
1440,4391, 399, 505,8596,6473,8169,4110,6660,7328,2895,7128, 225,
```

```
B4858,3296,2468,4764,8716,8075,9531,2268,2094,6447,1146,7141,3737,
C            -1,  21,   1,4351,4288,2843,2171, 936,5153,4591,8310,
D 752,9850,2618,6805,5957,7080,3259,5029, 670,4580,9720,1892,8967,
E6306,1110,6478,9732,2205,4882,3059,9882,2317,4848,7458,9287,8560,
F1378,9427,4487,5797,5194,1032,4036,1387,6588, 895,3621,9098,7433,
G             1,  21, 420, 905,7506,6658,6940,7247,2242,8288,5867,
H2330,6286,4274,3146, 729, 680,2763,3238,3912,4076,2551,2366,9039,
I6486,2654,5397,4051,7185,4894,4498,1050,3043,2694,7906,7426,6324,
J2645,1115,9387,3243, 736,6371,4052,4437,9099,4032,1370,1360,7079/
 DATA C11/   -1,  22,  12,7585,8127,2247,5759, 843,2594,5728,4945,
12045,3193,1983,  98,9091,7026,4530,6582,1855,8047,8594,3538,3133,
28557,3617,3869,7649,5244,4591,2821,2202,2217,8774,6568,3026,7234,
33312, 963,1430,4216, 795,1271, 218,7002,7416,7074,8635,5119,5921,
4             1,  22,4017,7568,4036,2181,8804,5903,1259,3158,7661,
55528,7105,4463, 534,9215,2322,3640,7637, 969,5373,1292,9279,3966,
65939,4848,6444,7253,8277,8480,1236,1286,6975,6988,3410,5913,7519,
73145,1046,4952,9428,2904,9023,7392,8922,6014,8897,3170,3890,9959,
8            -1,  23, 131,1013,6312,  68,3458,7473,7825,1695,5141,
97879,6212,6058,3407,3828,5327,8464,8866,2827,8032,4125,9381,8085,
A9568,5028,4498,5384,4264,3085,2370,4249,5701,3018,9142,7167,7720,
B4618, 299,6806,6814, 506, 181,7735,2001,9651,1913,5347,5804,4706,
C             1,  24,   4,4299,9565,3697,8259,6377,1152, 544,8820,
D5691,7838,4230,3837,6938, 826, 411,4234,9237,6322,4420,1146,7108,
E1628,3203,5794,7727,9202,2792, 227,9202,2792, 227,9202,2792, 227,
F9202,2792, 227,9202,2792, 227,9202,2792, 227,9202,2792, 227,9202,
G            -1,  24,1549,2140,6951,7355,3881,1542,8813,4953, 339,
H2547,1418,1057,3373,9601,2370,8171,7806,5382,5980,5723,1506,3202,
I2430,9461,8694,4563,3172,7198,5339, 496,4913,4777,4580,7540,9157,
J9200,4079,9201,4147, 582,2863,9475,2408,6179,3857,8358,8558,9707/
 DATA C12/   1,  25,  56, 376,4855,6273,5197,4950,6206,7876,8770,
18800,4366,9592, 900,6681,2345,8103,9077,2556, 238,1118,5400,4688,
29690,6256,8348,8783,8643,2506,8870,5234,1597,7961,4325, 688,7052,
33415,9779,6143,2506,8870,5234,1597,7961,4325, 688,7052,3415,9780,
4            -1,  26,   2, 953,9241,4858, 979,2706,4754,4170,7191,
52855,5944,3198,7352,4201,5953,2159,9949,5931,8851,1765,8764,7819,
65634,1485,8564,4000,5889, 834, 108,4010,8401, 840,1084, 108,4010,
78401, 840,1084, 108,4010,8401, 840,1084, 108,4010,8401, 840,1084,
8             1,  26, 809,5290,3005,6367,8084,  63,6463,6778,8443,
95997,2820,1881,8398,3044,5464, 364,1512,5125,8306,8966,6171, 379,
A6075,6370,3024,4291,7977,2386,6286, 197,7979,7685,8432,7087,3416,
B2790,4718,4731, 978,6496,7147,2187,3847,4849,8207,8470,  99,8163,
C            -1,  27,  32,2963,3556, 269,5213,4559,7210, 298,7523,
D 958,9812,3402,9130,9321,2400,3527,6342,4398,2642,2899,6957,3269,
E1956,9890,5726,7601,3392,4421,7179,5596,4373,9385,5179,8672,2247,
F9542,9983, 168,2877,8755,5967,2687,9728,2692,6046,  89,5476,3008,
G             1,  28,   1,3298,6391,7366,8232, 869,8228,8729,1823,
H 658,5413,4002, 922,8311,5936,1392,4138,9612,6198,2809,3169,9242,
I 583, 284,8428,4891,9580,9530,6273,6393,  42,4805,4842,7645,8890,
J3413,6859,4519,6685,4771,7544,6996,1464,4581,8024, 179,1750,2739/
 DATA C13/   -1,  28, 564,9116,4660,5107,5898,6024,4699,2770, 809,
12188,1037,6286,7139, 178,7531,9053, 990,9775,4518,4469,3938,9072,
26028,7011,3027,8752,3561,4806,3583,2703,1641,8990,6961,2807,5939,
39359,8917,5354,9089,1743,1518,4985,3654,4714,5494,7996,8980,5411,
4             1,  29,  24,7437,7846,3950,7093,8854,9333,8990, 862,
53718,9292,7520,2293,8585,6543,5842,8572, 530,4828,4601,9346,2012,
6 559,2257,2516,5851,8167,5704,1250, 319,7374,6867,1679,1979,9498,
77468,6716,7919,7994,9874,6867,1679,1979,9498,7468,6716,7919,7995,
```

```
     8            -1,  26,   4,1736, 993,9758,7346,5692,6499,2107,5293,
     9 959, 679,6456,7119,5997,1118,9800,3026,4559,5278,4030,6064,5006,
     A8446,9478,2023,6550,6493,9477,4104,3596,9352,1811,1195,2861,9528,
     B6195,2861,9528,6195,2861,9528,6195,2861,9528,6195,2861,9528,6195,
     C             1,  30, 519,4881,6925,7332,4717,4216,1057,8763,2244,
     D2830,2753,5169,1125, 769,9054,2487,8245,8918,7662, 710,8095,7382,
     E1990,4337,5511,1986,2829,4235,7561, 914,1540,4992, 635,4526,7883,
     F4901,1514,7249,4491,2616,4868, 301,4269,9675,5933,1719,5022,6299,
     G            -1,  31,  24,8780, 657,6674,6559,6546,2074,3627,7667,
     H6650,1184,7376,2756,7422,3284,4736,  10,2910,1493,5876,2875,2404,
     I 645,1038,8936,8521,6462,1414,5918,4237,5255,2276,9388, 641,3653,
     J1297,7222, 900,4839, 445,6636,9069, 787, 321,6948,7157,4152,2196/
      DATA C14/    1,  32,   1,2263, 550,4019,6275,4019,4958,9153,9749,
     17272,9834, 300,8689,6615,9526,1452,3188,9983,9015,5480,3408,8050,
     26414,7680,9821,5521,4343,2515,4458,7641,5660,1133,2410,5916, 756,
     35011,8203,3096,9267,1394,7990,5437,3522,4586,2884,1607,5650,1182/
      DATA LN2PI/   1,  -1,9189,3853,3204,6727,4178, 329,7364, 561,7639,
     1 8613,9747,3637,7834,1281,7151,5404,8276,5695,9272,6039,7694,7432,
     2 9863,5954,1976,2200,5646,6246,3433,7446,3668,6288,1840,7935,7215,
     3 5875,9152,2268,1393,6035,6074,2547,3586,6904,6395,9059,9138,0806/
      IF(ZIN(1).GT.0) GO TO 4
      WRITE(6,1000)
1000  FORMAT('- ***ERROR IN LGAM*** Z IS LESS THAN OR EQUAL TO 0 WHEN CA
     1LCULATING GAMMA(X)')
      RETURN
4     CALL COPY(ZIN,Z,K)
      SW = 0
      CALL SET(ONE,1,&5,K)
    5 IT = D - 10
      IEND=20
      IF (D.GT.50)IEND = 31
      IF (D.GT.80)IEND = 41
      IF (D.GT.110)IEND = 51
      IF(D.GT.137) IEND=61
      IF (D.GT.165)IEND = 71
      IF (D.GT.16)GO TO 10
      IT = 7
   10 CALL SET(T,IT,&20,K)
   20 CALL SBTRCT(Z,T,TEMP2,K)
      IF (TEMP2(1).GE.0)GO TO 60
C
C *** SCALE Z SO CALCULATION IS LN GAMMA(Z) = LN GAMMA(Z+K) - LN(PRODUCT
C ***   AS I=0,K-1 OF (Z+I))
      CALL COPY(Z,TEMP,K)
   40 CALL ADD(Z,ONE,Z,K)
      CALL ADD(TEMP2,ONE,TEMP2,K)
      IF (TEMP2(1).GE.0)GO TO 50
      CALL MLTPLY(TEMP,Z,TEMP,K)
      GO TO 40
   50 SW = 1
C
C *** EVALUATE LN GAMMA(Z) BY STIRLING ASYMPTOTIC SERIES
C
C
C *** CALCULATION BEGINS HERE
   60 CALL COPY(Z,TEMP2,K)
      TEMP3(1) = -1
```

```
      TEMP3(2) = -1
      TEMP3(3) = 5000
      DO 410 I=4,K
  410 TEMP3(I) = 0
      CALL ADD(TEMP2,TEMP3,TEMP2,K)
      CALL LNGLOG(Z,TEMP3,&420,K)
  420 CALL MLTPLY(TEMP2,TEMP3,TEMP3,K)
      CALL SBTRCT(TEMP3,Z,TEMP3,K)
      CALL ADD(TEMP3,LN2PI,TEMP3,K)
C
C *** TEMP3 NOW = (Z-.5)*LN(Z)-Z+LN(SQRT(2*PI))
      CALL COPY(Z,TEMP2,K)
      CALL MLTPLY(Z,Z,Z,K)
C
C *** THIS LOOP DOES LN GAMMA(Z)=TEMP3+SUMMATION(C(I)/Z**(2I-1))
      DO 430 I=1,IEND
      CALL DIVIDE(C(1,I),TEMP2,Y,K)
      CALL ADD(Y,TEMP3,Y,K)
      CALL SBTRCT(Y,TEMP3,TEMP3,K)
      IF (TEMP3(1).EQ.0)GO TO 440
      CALL MLTPLY(TEMP2,Z,TEMP2,K)
      CALL COPY(Y,TEMP3,K)
  430 CONTINUE
  440 IF (SW.EQ.0)GO TO 460
C
C *** CORRECTION FOR SCALING OF Z
      CALL LNGLOG(TEMP,TEMP,&450,K)
  450 CALL SBTRCT(Y,TEMP,Y,K)
C
C *** GAMMA = EXP(LN GAMMA)
  460 CALL LEXP(Y,Y,K)
      RETURN
      END
```

ENTRY NAME
ILEG1

IDENTIFICATION *** Multiple Precision Associated Legendre Function of the First Kind for Integer Order and Degree.

PURPOSE *** This subroutine generates, in multiple precision, the associated Legendre functions of the first kind P(X), for order N=0,1,...,NX-1 and integer degree A.GE.0 and X.GE.1.

OTHER SUBROUTINES USED *** IBCOM, LNGCNS, SET, SBTRCT, COPY, MLTPLY, LSQRT, MLTINT, ADD, EXPO and DIVIDE.

USAGE *** INTEGER X(KX),A,NX,P(KX,NXX),K,KX,RR(KX,NXX)

```
      .
      .
      .
      CALL ILEG1(X,A,NX,P,K,&L,KX,RR)
```

where the parameters in the calling sequence are:

X - The multiple precision argument stored in the form X.EQ.A*(10**4)**B where 1.LE.A.LT.10**4 as follows:

X(1) = sign of the number: +1 if positive
-1 if negative
0 if zero (In this case, X(2) to X(K) are disregarded)

X(2) = exponent B of 10**4

X(3) = integral part of the mantissa, 1.LE.X(3).LT.10**4

X(4) to X(K) = fractional part of the mantissa, stored 4 decimal digits per location.

The value of X must be greater than or equal to one.

input

A - The integer degree of the Legendre functions which will be generated. A.GE.0

input

NX - The Legendre functions will be generated for order 0,1,..., NX-1. NX.GT.0, NX.LE.NXX

input

P - On output, the Jth column, P(I,J), I=1, ..., K will contain the multiple precision Legendre function of degree A and order J-1 stored in the same manner as X.

output

K - The number of locations used in the arrays to store the multiple precision numbers. K.LE.KX.LE.50

input

L - Control returns immediately to the statement labelled L (an actual integer) if X.LT.1 or A.LT.0 or NX.LE.0. Also see ERROR MESSAGES.

input

KX - The maximum value of K. KX.LE.50

input

RR - An array used for temporary storage in ILEG1.

ERROR MESSAGES *** If the parameters A and NX are too large the program will be unable to continue execution and control will return to the label parameter L specified in the calling sequence after the following message has been printed:

*** ERROR IN ILEG1 *** NUMBERS HAVE BECOME TOO LARGE

ACCURACY *** In test cases there were (K-4)*4+3 significant digits in the generated function values. If X is very close to 1 there will probably be a loss of accuracy.

```
      SUBROUTINE ILEG1(X,A,NX,P,K,*,KX, RR)
C *** THIS SUBROUTINE GENERATES, IN MULTIPLE PRECISION, THE ASSOCIATED
C ***   LEGENDRE FUNCTIONS OF THE FIRST KIND P(X), FOR ORDER N=0,1,...,
C ***   NX-1 AND INTEGER DEGREE M.GE.0 AND X.GE.1
      COMMON /LONG/LN,LW,LT,LS,LR,LTR,LEX,LRM
      INTEGER LN,LW,LT,LS,LEX
      REAL LR,LTR,LRM
      INTEGER AP2,END,KX, AP1MN
      INTEGER X(1),A,NX,P( KX,1),N,C(50),X1(50),SUM(50),R(50),S(50),
     1        RR( KX,1), TEMP(50),TEMP2(50),ZERO(50),ONE(50),TWO(50)
      CALL LNGCNS(K,4,8,75)
C *** INITIALIZE CONSTANTS FOR LONG ARITHMETIC
      CALL SET(ZERO,0,&10,K)
   10 CALL SET(ONE,1,&20,K)
   20 CALL SET(TWO,2,&30,K)
C
C *** CHECK IF INPUT PARAMETERS ARE IN CORRECT RANGE
30    CALL SBTRCT(X,ONE,TEMP,K)
      IF(TEMP(1).LT.0.OR.A.LT.0.OR.NX.LE.0) RETURN1
C
C
C *** IF X=1  OR A=0 THEN P(0)=1 AND P(N)=0, N=1,NMAX
      IF (TEMP(1).NE.0.AND.A.NE.0) GO TO 50
      CALL COPY(ONE,P(1,1),K)
      IF (NX.LT.2)GO TO 45
      DO 40 N=2,NX
      CALL COPY(ZERO,P(1,N),K)
   40 CONTINUE
   45 RETURN
C
C *** P(N)=0 FOR N=A+1,NMAX
   50 IF (A+2.GT.NX)GO TO 65
      AP2 = A + 2
      DO 60 N=AP2,NX
      CALL COPY(ZERO,P(1,N),K)
   60 CONTINUE
C
C *** X1 = SQRT(X**2-1)
   65 CALL MLTPLY(X,X,TEMP,K)
      CALL SBTRCT(TEMP,ONE,TEMP,K)
      CALL LSQRT(TEMP,X1,K)
C
C *** C = A FACTORIAL
      CALL COPY(ONE,C,K)
      IF (A.LT.2)GO TO 95
      DO 90 N=2,A
      CALL MLTINT(C,N,C,&300,K)
   90 CONTINUE
C
C *** SUM = (X + X1)**A / C
   95 CALL ADD(X,X1,TEMP,K)
      CALL EXPO(TEMP,A,TEMP,K)
      CALL DIVIDE(TEMP,C,SUM,K)
C
C *** X1 = 2 * X / X1
      CALL MLTINT(X,2,TEMP,&100,K)
  100 CALL DIVIDE(TEMP,X1,X1,K)
```

```
C
C *** R = S = 0
      CALL COPY(ZERO,R,K)
      CALL COPY(ZERO,S,K)
      DO 150 NN=1,A
      N = A + 1 - NN
C
C *** R = (A+1-N) / (N*X1 + (N+A+1) * R)
      CALL MLTINT(X1,N,TEMP,&300,K)
  110 NPAP1 = N + A + 1
      CALL MLTINT(R,NPAP1,TEMP2,&300,K)
  120 CALL ADD(TEMP,TEMP2,TEMP,K)
      AP1MN = A + 1 - N
      CALL SET(TEMP2,AP1MN,&300,K)
  140 CALL DIVIDE(TEMP2,TEMP,R,K)
C
C *** S = R * (2 + S)
      CALL ADD(S,TWO,TEMP,K)
      CALL MLTPLY(R,TEMP,S,K)
      IF (N.GE.NX)GO TO 150
C
C *** RR(N-1) = R
      CALL COPY(R,RR(1,N),K)
  150 CONTINUE
C
C *** P(0) = C * SUM / (1 + S)
      CALL MLTPLY(C,SUM,TEMP,K)
      CALL ADD(ONE,S,TEMP2,K)
      CALL DIVIDE(TEMP,TEMP2,P(1,1),K)
      IF (NX.GT.A+1)GO TO 160
      END = NX-1
      GO TO 170
  160 END = A
  170 DO 200 N=1,END
      NP1 = N+1
C
C *** P(N+1) = (N+A+1) * RR(N) * P(N)
      CALL MLTPLY(P(1,N),RR(1,N),TEMP,K)
      NPAP1 = NP1 + A - 1
      CALL MLTINT(TEMP,NPAP1,P(1,NP1),&300,K)
  200 CONTINUE
      RETURN
C
C *** ERROR RETURN
  300 WRITE(6,1000)
 1000 FORMAT('-  *** ERROR IN ILEG1 *** NUMBERS HAVE BECOME TOO LARGE')
      RETURN1
      END
```

ENTRY NAME
ILEG2

IDENTIFICATION *** Multiple Precision Associated Legendre Function
of the Second Kind for Integer Order and Degree.

PURPOSE *** This subroutine generates, in multiple precision, to D significant digits the associated Legendre function of the second kind Q(X), for integer order M.GE.0 and degree N=0,1,...,NX-1 and X.GT.1.

OTHER SUBROUTINES USED *** FIXPI, IBCOM, LNGCNS, SET, SBTRCT, MLTPLY, LSQRT, ADD, DIVIDE, LNGLOG, COPY, RECIP and MLTINT.

USAGE *** INTEGER X(KX),M,NX,D,Q(KX,NXX),K,KX,QQ(KX,NXX),RR(KX,NXX)

```
      :
      :
     CALL ILEG2(X,M,NX,D,Q,K,&L,KX,QQ,RR)
```

where the parameters in the calling sequence are:

X - The multiple precision argument stored in the form X.EQ.A*(10**4)**B where 1.LE.A.LT.10**4 as follows:

X(1) = sign of the number: +1 if positive
-1 if negative
0 if zero (In this case, X(2) to X(K) are disregarded)

X(2) = exponent B of 10**4

X(3) = integral part of the mantissa, 1.LE.X(3).LT.10**4

X(4) to X(K) = fractional part of the mantissa, stored 4 decimal digits per location.

The value of X must be greater than one.

input

M - The integer order of the Legendre functions which will be generated. M.GE.0

input

NX - The Legendre functions will be generated for degree 0, 1, ..., NX-1. NX.GT.0, NX.LE.NXX

input

D - The number of significant digits desired in the function values. A suggested value to use is (K-4)*4+3.

input

Q - On output, the Jth column, Q(I,J), I=1, ..., K will contain the multiple precision Legendre function of order M and degree J-1 stored in the same manner as X.

output

K - The number of locations used in the arrays to store the multiple precision numbers. K.LE.KX.LE.50

input

L - Control returns immediately to the statement labelled L (an actual integer) if X.LE.1 or M.LT.0 or NX.LE.0. Also see ERROR MESSAGES.

input

KX - The maximum value of K.KX.LE.50

input

QQ,RR - Arrays used for temporary storage in ILEG2.

ERROR MESSAGES *** In some cases the program will be unable to continue execution and control will return to the label parameter L

specified in the calling sequence after the following message has been printed:

*** ERROR IN ILEG2 *** NUMBERS HAVE BECOME TOO LARGE

This can occur if M or NX are too large or if the number of significant digits D asked for is too great and the algorithm is failing to converge.

ACCURACY *** There should be D significant digits in the generated function values. However if D is set equal to (K-3)*4 there may only be D-1 significant digits. Convergence is slow for X near 1.

```
      SUBROUTINE ILEG2(X,M,NX,D,Q,K,*,KX, QAPRX,RR)
C *** THIS SUBROUTINE GENERATES, IN MULTIPLE PRECISION, TO D SIGNIFICANT
C *** DIGITS THE ASSOCIATED LEGENDRE FUNCTIONS OF THE SECOND KIND Q(X),
C *** FOR INTEGER ORDER M.GE.0 AND DEGREE N=0,1,...,NX-1 AND X.GT.1
      INTEGER X(1),M,NX,D,K,Q(KX,1),N,NU,NN,QAPRX(KX,1),RR(KX,1),
     1         X1(50),Q0(50),Q1(50),Q2(50),R(50),EPS(50),TEMP(50),
     2         TEMP2(50),ZERO(50),ONE(50),HALF(50)
      CALL LNGCNS(K,4,8,75)
C
C *** INITIALIZE CONSTANTS FOR LONG ARITHMETIC
      CALL SET(ZERO,0,&20,K)
   20 CALL SET(ONE,1,&30,K)
   30 HALF(1) = 1
      HALF(2) = -1
      HALF(3) = 5000
      DO 35 I=4,K
   35 HALF(I) = 0
C
C *** CHECK IF INPUT PARAMETERS IN CORRECT RANGE
      CALL SBTRCT(X,ONE,TEMP,K)
      IF (TEMP(1).LE.0.OR.NX.LE.0.OR.M.LT.0)RETURN 1
C
C *** X1 = SQRT(X**2-1)
   40 CALL MLTPLY(X,X,TEMP,K)
      CALL SBTRCT(TEMP,ONE,TEMP,K)
      CALL LSQRT(TEMP,X1,K)
C
C *** Q1 = .5 * LN((X+1)/(X-1))
      CALL ADD(X,ONE,TEMP,K)
      CALL SBTRCT(X,ONE,TEMP2,K)
      CALL DIVIDE(TEMP,TEMP2,TEMP,K)
      CALL LNGLOG(TEMP,TEMP,&50,K)
   50 CALL MLTPLY(TEMP,HALF,Q1,K)
      IF (M.NE.0)GO TO 60
C
C *** Q(0) = Q1
      CALL COPY(Q1,Q(1,1),K)
      GO TO 160
C
C *** Q2 = -1/X1
   60 CALL RECIP(X1,Q2,K)
      Q2(1) = -1 * Q2(1)
C
C *** X1 = 2*X/X1
      CALL MLTINT(X,2,TEMP,&80,K)
   80 CALL MLTPLY(TEMP,Q2,X1,K)
      X1(1) = -1 * X1(1)
      MM1 = M-1
      IF (MM1.EQ.0)GO TO 155
      DO 150 N=1,MM1
C
C *** Q0 = Q1
      CALL COPY(Q1,Q0,K)
C
C *** Q1 = Q2
      CALL COPY(Q2,Q1,K)
C
```

```
*** Q2 = -N*X1*Q1-N*(N-1)*Q0
    NNM1 = N * (N-1)
    CALL MLTINT(Q0,NNM1,TEMP2,&500,K)
    CALL MLTINT(X1,-N,TEMP,&500,K)
    CALL MLTPLY(TEMP,Q1,TEMP,K)
    CALL SBTRCT(TEMP,TEMP2,Q2,K)
150 CONTINUE

*** Q(0) = Q2
155 CALL COPY(Q2,Q(1,1),K)
160 DO 170 N=1,NX

*** QAPRX(N) = 0
    CALL COPY(ZERO,QAPRX(1,N),K)
170 CONTINUE

*** EPS = .5 * 10**(-D)
    EPS(1) = +1
    EPS(2) = -D/4 - 1
    EPS(3) = 5 * 10**(-4*EPS(2) - D - 1)
    DO 180 I=4,K
180 EPS(I) = 0
    NU = 20 +  IFIX(1.25*(NX-1))
    NUINC = 10

*** R = 0
210 CALL COPY(ZERO,R,K)
    DO 250 NN=1,NU
    N = NU-NN+1

*** R = (N+M)/((2*N+1)*X-(N-M+1)*R)
    N2P1 = 2*N+1
    CALL MLTINT(X,N2P1,TEMP,&500,K)
    NM1 = N-M+1
    CALL MLTINT(R,NM1,TEMP2,&500,K)
    CALL SBTRCT(TEMP,TEMP2,TEMP2,K)
    NM = N+M
    CALL RECIP(TEMP2,TEMP2,K)
    CALL MLTINT(TEMP2,NM,R,&500,K)
    IF (N.GE.NX)GO TO 250

*** RR(N-1) = R
    CALL COPY(R,RR(1,N),K)
250 CONTINUE
    NXM1 = NX-1

*** Q(N+1) = RR(N) * Q(N)
    DO 260 N=1,NXM1
    CALL MLTPLY(RR(1,N),Q(1,N),Q(1,N+1),K)
260 CONTINUE
    DO 280 N=1,NX

*** IF(ABS(Q(N)-QAPRX(N)).GT.EPS*ABS(Q(N))GO TO 290
    CALL SBTRCT(Q(1,N),QAPRX(1,N),TEMP,K)
    TEMP(1) = +1
    CALL MLTPLY(EPS,Q(1,N),TEMP2,K)
    TEMP2(1) = +1
```

```
      CALL SBTRCT(TEMP,TEMP2,TEMP,K)
      IF(TEMP(1).GT.0) GO TO 290
  280 CONTINUE
      RETURN
  290 DO 300 N=1,NX
C
C *** QAPRX(N) = Q(N)
      CALL COPY(Q(1,N),QAPRX(1,N),K)
  300 CONTINUE
      NU = NU + NUINC
      NUINC = 2 * NUINC
      GO TO 210
C
C *** ERROR RETURN
  500 WRITE (6,100)
 100 FORMAT ('- *** ERROR IN ILEG2 *** NUMBERS HAVE BECOME TOO LARGE')
      RETURN1
      END
```

ENTRY NAME
ILEG3

IDENTIFICATION *** Multiple Precision Associated Legendre Function of the Second Kind for Integer Order and Degree.

PURPOSE *** This subroutine generates, in multiple precision, to D significant digits the associated Legendre functions of the second kind Q(X), for order M=0,1,...,MX-1 and integer degree N.GE.0 and X.GT.1.

OTHER SUBROUTINES USED *** IBCOM, ILEG2, COPY, MLTPLY, SET, SBTRCT, LSQRT, MLTINT and DIVIDE.

USAGE *** INTEGER X(KX),N,MX,D,Q(KX,MXX),KX,QQ(KX,MXX),RR(KX,MXX),K

```
      .
      .
      .
      CALL ILEG3(X,N,MX,D,Q,K,&L,KX,QQ,RR)
```

where the parameters in the calling sequence are:

X - The multiple precision argument stored in the form X.EQ.A*(10**4)**B where 1.LE.A.LT.10**4 as follows:

X(1) = sign of the number: +1 if positive
-1 if negative
0 if zero (In this case, X(2) to X(K) are disregarded)

X(2) = exponent B of 10**4

X(3) = integral part of the mantissa, 1.LE.X(3).LT.10**4

X(4) to X(K) = fractional part of the mantissa, stored 4 decimal digits per location.

The value of X must be greater than one.

input

N - The integer degree of the Legendre functions which will be generated. N.GE.0

input

MX - The Legendre functions will be generated for order 0,1,...,MX-1. MX.GT.0, MX.LE.MXX

input

D - The number of significant digits desired in the function values. A suggested value to use is (K-4)*4+3.

input

Q - On output, the Jth column, Q(I,J), I=1,...K will contain the multiple precision Legendre function of degree N and order J-1 stored in the same manner as X.

output

K - The number of locations used in the arrays to store the multiple precision numbers. K.LE.KX.LE.50

input

L - Control returns immediately to the statement labelled L (an actual integer) if X.LE.1 or N.LT.0 or MX.LE.0. Also see ERROR MESSAGES.

input

KX - The maximum value of K. KX.LE.50

input

QQ, RR - Arrays used for temporary storage in ILEG3.

ERROR MESSAGES *** If the parameters N and MX are too large the program will be unable to continue execution and control will return to the label parameter L specified in the calling sequence after the following message has been printed:

*** ERROR IN ILEG3 *** NUMBERS HAVE BECOME TOO LARGE

Also see ERROR MESSAGES in the description of ILEG2.

ACCURACY *** There should be D significant digits in the generated function values. However if D is set equal to (K-3)*4 there may only be D-1 significant digits. Convergence is slow for X near 1.

```
      SUBROUTINE ILEG3(X,N,MX,D,Q,K,*,KX,QAPRX,RR)
C *** THIS SUBROUTINE GENERATES, IN MULTIPLE PRECISION, TO D SIGNIFICANT
C *** DIGITS THE ASSOCIATED LEGENDRE FUNCTIONS OF THE SECOND KIND Q(X),
C *** FOR ORDER M=0,1,...,MX-1 AND INTEGER DEGREE N.GE.0 AND X.GT.1
      COMMON /LONG/LN,LW,LT,LS,LR,LTR,LEX,LRM
      INTEGER LN,LW,LT,LS,LEX
      REAL LR,LTR,LRM
      INTEGER X(1),N,MX,D,K,KX,Q(KX,1),QAPRX(KX,1),RR(KX,1),
     1        M,X1(50),TEMP(50),TEMP2(50)
C *** CHECK IF INPUT PARAMETERS IN CORRECT RANGE
      IF (N.LT.0.OR.MX.LT.1)RETURN 1
      CALL ILEG2(X,0,N+1,D,Q,K,&60,KX,QAPRX,RR)
      CALL COPY(Q(1,N+1),TEMP,K)
      IF (MX.GT.1)GO TO 10
      CALL COPY(TEMP,Q(1,1),K)
      RETURN
   10 CALL ILEG2(X,1,N+1,D,Q,K,&60,KX,QAPRX,RR)
      CALL COPY(Q(1,N+1),Q(1,2),K)
      CALL COPY(TEMP,Q(1,1),K)
      IF (MX.EQ.2)RETURN
C
C *** X1 = 2*X/SQRT(X**2-1)
      CALL MLTPLY(X,X,TEMP,K)
      CALL SET(TEMP2,1,&20,K)
   20 CALL SBTRCT(TEMP,TEMP2,TEMP,K)
      CALL LSQRT(TEMP,TEMP2,K)
      CALL MLTINT(X,2,TEMP,&25,K)
   25 CALL DIVIDE(TEMP,TEMP2,X1,K)
      MXM1 = MX-1
      DO 50 M=2,MXM1
C
C *** Q(M+1) = -M*X1*Q(M)-(M+N)*(M-N-1)*Q(M-1)
      CALL MLTINT(X1,-(M-1),TEMP,&100,K)
      CALL MLTPLY(Q(1,M),TEMP,TEMP,K)
      MM = ((M-1)+N) * ((M-1)-N-1)
      CALL MLTINT(Q(1,M-1),MM,TEMP2,&100,K)
      CALL SBTRCT(TEMP,TEMP2,Q(1,M+1),K)
   50 CONTINUE
      RETURN
   60 RETURN1
C
C *** ERROR RETURN
  100 WRITE (6,1000)
 1000 FORMAT ('- *** ERROR IN ILEG3 *** NUMBERS HAVE BECOME TOO LARGE')
      RETURN1
      END
```

ENTRY NAME
LEG1

IDENTIFICATION *** Multiple Precision Associated Legendre Function of the First Kind for Integer Order and Non-integer Degree.

PURPOSE *** This subroutine generates, in multiple precision, to D significant digits the associated Legendre functions of the first kind P(X), for order N=0,1,...,NX-1 and non-integer degree A and X.GE.1.

OTHER SUBROUTINES USED *** FIXPI, IBCOM, LNGCNS, SET, SBTRCT, COPY, ADD, LGAM, MLTPLY, LSQRT, LEXPO, DIVIDE, MLTINT and SHORTN.

USAGE *** INTEGER X(KX),A(KX),NX,D,P(KX,NXX),K,KX,PP(KX,NXX),RR(KX,NXX)

```
      :
      :
      CALL LEG1(X,A,NX,D,P,K,&L,KX,PP,RR)
```

where the parameters in the calling sequence are:

X - The multiple precision argument stored in the form X.EQ.A*(10**4)**B where 1.LE.A.LT.10**4 as follows:

X(1) = sign of the number: +1 if positive
-1 if negative
0 if zero (In this case, X(2) to X(K) are disregarded)

X(2) = exponent B of 10**4

X(3) = integral part of the mantissa, 1.LE.X(3).LT.10**4

X(4) to X(K) = fractional part of the mantissa, stored 4 decimal digits per location.

The value of X must be greater than or equal to one.

input

A - The multiple precision degree, stored in the same manner as X, of the Legendre functions which will be generated. A must not be an integer.

input

NX - The Legendre functions will be generated for order 0,1,...,NX-1. NX.GT.0, NX.LE.NXX

input

D - The number of significant digits desired in the function values. A suggested value to use is (K-4)*4+1.

input

P - On output, the Jth column, P(I,J), I=1,...,K will contain the multiple precision Legendre function of degree A and order J-1 stored in the same manner as X.

output

K - The number of locations used in the arrays to store the multiple precision numbers. K.LE.KX.LE.50

input

L - Control returns immediately to the statement labelled L (an actual integer) if X.LT.1 or NX.LE.0 or A is an integer. Also see ERROR MESSAGES.

input

KX - The maximum value of K. KX.LE.50

input

PP, RR - Arrays used for temporary storage in LEG1.

ERROR MESSAGES *** In some cases the program will be unable to continue execution and control will return to the label parameter L specified in the calling sequence after the following message has been printed:

*** ERROR IN LEG1 *** NUMBERS HAVE BECOME TOO LARGE

This can occur if A or NX are too large or if the number of significant digits D asked for is too great and the algorithm is failing to converge.

ACCURACY *** In all cases tested there were D significant digits in the generated function values. However accuracy may be affected if X is very close to 1. The rate of convergence of this algorithm decreases as X increases.

```
      SUBROUTINE LEG1(X,ALPH,NX,D,P1,K,*,KX,PAPRX,RR)
C *** THIS SUBROUTINE GENERATES, IN MULTIPLE PRECISION, TO D SIGNIFICANT
C *** DIGITS THE ASSOCIATED LEGENDRE FUNCTIONS OF THE FIRST KIND P(X),
C *** FOR ORDER N=0,1,...,NX-1 AND NON-INTEGER DEGREE ALPH AND X.GE.1
      COMMON /LONG/LN,LW,LT,LS,LR,LTR,LEX,LRM
      INTEGER LN,LW,LT,LS,LEX
      REAL LR,LTR,LRM
      INTEGER X(1),ALPH(1),NX,D,K,KX,P1(KX,1),PAPRX(KX,1),RR(KX,1),N,NU,
     1        M,A(50),EPS(50),X1(50),SUM(50),C(50),R(50),S(50),ZERO(50),
     2        ONE(50),TWO(50),TEMP(50),TEMP2(50)
      CALL LNGCNS(K,4,8,75)
C
C *** INITIALIZE CONSTANTS FOR LONG ARITHMETIC
      CALL SET(ZERO,0,&10,K)
   10 CALL SET(ONE,1,&20,K)
   20 CALL SET(TWO,2,&30,K)
C
C *** CHECK IF INPUT PARAMETERS ARE IN CORRECT RANGE
   30 CALL SBTRCT(X,ONE,TEMP,K)
      IF (TEMP(1).LT.0.OR.NX.LE.0)RETURN 1
      IF (ALPH(2).LT.0)GO TO 50
      L = ALPH(2) + 3
      IF (L.GT.K)GO TO 50
      DO 40 I=L,K
      IF (ALPH(I).NE.0)GO TO 50
   40 CONTINUE
   45 RETURN 1
   50 IF (TEMP(1).NE.0)GO TO 70
C
C *** IF X=1 THEN P1(0)=1 AND P1(N)=0, N=1,NMAX
      CALL COPY(ONE,P1(1,1),K)
      IF (NX.LT.2)RETURN
      DO 60 N=2,NX
      CALL COPY(ZERO,P1(1,N),K)
   60 CONTINUE
      RETURN
C
C *** A=IF ALPHA.LT.-.5 THEN -ALPHA-1 ELSE ALPHA
   70 TEMP(1) = -1
      TEMP(2) = -1
      TEMP(3) = 5000
      DO 75 I=1,K
   75 TEMP(I) = 0
      CALL SBTRCT(ALPH,TEMP,TEMP,K)
      IF (TEMP(1).GE.0)GO TO 80
      CALL ADD(ALPH,ONE,A,K)
      A(1) = -A(1)
      GO TO 90
   80 CALL COPY(ALPH,A,K)

C *** PAPROX(N)=0 FOR N=0,NMAX
   90 DO 100 N=1,NX
      CALL COPY(ZERO,PAPRX(1,N),K)
  100 CONTINUE

C *** EPSILON = .5 *10**(-D)
      EPS(1) = +1
```

```
      EPS(2) = -D/4 - 1
      EPS(3) = 5 * 10**(-4*EPS(2) - D - 1)
      DO 110 I=4,K
  110 EPS(I) = 0
C
C *** C = GAMMA(1+A)
      CALL ADD(A,ONE,TEMP,K)
      CALL LGAM(TEMP,C,D,K)
C
C *** X1 = SQRT(X**2 - 1)
      CALL MLTPLY(X,X,TEMP,K)
      CALL SBTRCT(TEMP,ONE,TEMP,K)
      CALL LSQRT(TEMP,X1,K)
C
C *** SUM = (X+X1)**A/C
      CALL ADD(X,X1,TEMP,K)
      CALL LEXPO(TEMP,A,TEMP,K)
      CALL DIVIDE(TEMP,C,SUM,K)
C
C *** X1 = 2*X / X1
      CALL MLTINT(X,2,TEMP,&180,K)
  180 CALL DIVIDE(TEMP,X1,X1,K)
C
C *** NU=20+ENTIER((37.26+.1283*(A+38.26)*X)*NMAX/(37.26+.1283*(A+1)*X))
      CALL SHORTN(X,XS,&45,K)
      CALL SHORTN(A,AS,&45,K)
      NU = 20 + AINT((37.26+.1283*(AS+38.26)*XS)*(NX-1)/(37.26+.1283*
     1                (AS+1)*XS))
      NUINC = 10
C
C *** R = S = 0
  190 CALL COPY(ZERO,R,K)
      CALL COPY(ZERO,S,K)
      DO 210 NN=1,NU
      N = NU-NN+1
C
C *** R=(A+1-N)/(N*X1+(N+A+1)*R)
      CALL SET(TEMP,N,&500,K)
      CALL ADD(TEMP,A,TEMP2,K)
      CALL ADD(TEMP2,ONE,TEMP2,K)
      CALL MLTPLY(TEMP2,R,R,K)
      CALL MLTPLY(TEMP,X1,TEMP2,K)
      CALL ADD(TEMP2,R,R,K)
      CALL ADD(A,ONE,TEMP2,K)
      CALL SBTRCT(TEMP2,TEMP,TEMP,K)
      CALL DIVIDE(TEMP,R,R,K)
C
C *** S = R*(2+S)
      CALL ADD(S,TWO,TEMP,K)
      CALL MLTPLY(TEMP,R,S,K)
      IF (N.GE.NX)GO TO 210
C
C *** IF N.LE.NMAX THEN RR(N-1)=R
      CALL COPY(R,RR(1,N),K)
  210 CONTINUE
C
C *** P1(0) = SUM / (1+S)
```

```
      CALL ADD(S,ONE,TEMP,K)
      CALL DIVIDE(SUM,TEMP,P1(1,1),K)
C
C *** P1(N+1) = RR(N)*P1(N) FOR N=0,NMAX-1
      IF (NX.EQ.1)GO TO 230
      NXM1 = NX-1
      DO 220 N=1,NXM1
      CALL MLTPLY(RR(1,N),P1(1,N),P1(1,N+1),K)
  220 CONTINUE
  230 DO 240 N=1,NX
C
C *** IF (ABS(P1(N)-PAPPROX(N)).GT.EPSILON*ABS(P1(N))GO TO 270
      CALL SBTRCT(P1(1,N),PAPRX(1,N),TEMP,K)
      TEMP(1) = +1
      CALL MLTPLY(EPS,P1(1,N),TEMP2,K)
      TEMP2(1) = +1
      CALL SBTRCT(TEMP,TEMP2,TEMP,K)
      IF (TEMP(1).GT.0)GO TO 270
  240 CONTINUE
C
C *** P1(0) = C*P1(0)
      CALL MLTPLY(C,P1(1,1),P1(1,1),K)
      IF (NX.EQ.1)GO TO 265
      DO 260 NN=2,NX
      N = NN-1
C
C *** C = (A+N)*C
      CALL SET(TEMP,N,&500,K)
      CALL ADD(TEMP,A,TEMP,K)
      CALL MLTPLY(TEMP,C,C,K)
C
C *** P1(N) = C * P1(N)
      CALL MLTPLY(C,P1(1,NN),P1(1,NN),K)
  260 CONTINUE
  265 RETURN
C
C *** PAPPROX(M) = P1(M) FOR M=0,NMAX
  270 DO 280 N=1,NX
      CALL COPY(P1(1,N),PAPRX(1,N),K)
  280 CONTINUE
      NU = NU + NUINC
      NUINC = 2 * NUINC
      GO TO 190
C
C *** ERROR RETURN
  500 WRITE (6,1000)
 1000 FORMAT ('- *** ERROR IN LEG1 *** NUMBERS HAVE BECOME TOO LARGE')
      RETURN 1
      END
```

ENTRY NAME
LEG2

IDENTIFICATION *** Multiple Precision Associated Legendre Function of the First Kind for Integer Order and Non-integer Degree.

PURPOSE *** This subroutine generates, in multiple precision, to D significant digits the associated Legendre functions of the first kind P(X), for integer order M and degree A, A+1, A+2,...,A+NX-1, A not an integer, and X.GE.1.

OTHER SUBROUTINES USED *** IBCOM, LEG1, COPY, SET, ADD, MLTPLY, SBTRCT and DIVIDE.

USAGE *** INTEGER X(KX),A(KX),M,NX,D,P(KX,NNX),K,KX,PP(KX,NXX),RR(KX,NXX)

```
      :
      :
      CALL LEG2(X,A,M,NX,D,P,K,&L,KX,PP,RR)
```

where the parameters in the calling sequence are:

X - The multiple precision argument stored in the form X.EQ.A*(10**4)**B where 1.LE.A.LT.10**4 as follows:

X(1) = sign of the number: +1 if positive
-1 if negative
0 if zero (In this case, X(2) to X(K) are disregarded)

X(2) = exponent B of 10**4

X(3) = integral part of the mantissa, 1.LE.X(3).LT.10**4

X(4) to X(K) = fractional part of the mantissa, stored 4 decimal digits per location.

The value of X must be greater than or equal to one.

input

A - The multiple precision degree, stored in the same manner as X, of the first Legendre function which will be generated. A must not be an integer.

input

M - The integer order of the Legendre functions generated. M.GE.0

input

NX - The Legendre functions will be generated for degree A, A+1, A+2, ..., A+NX-1. NX.GT.0, NX.LE.NXX

input

D - The number of significant digits desired in the function values. A suggested value to use is (K-4)*4+1.

input

P - On output, the Jth column, P(I,J), I=1,...,K will contain the multiple precision Legendre function of order M and degree A+J-1 stored in the same manner as X.

output

K - The number of locations used in the arrays to store the multiple precision numbers. K.LE.KX.LE.50

input

L - Control returns immediately to the statement labelled L (an actual integer) if X.LT.1 or NX.LE.0 or M.LT.0 or A is an integer. Also see ERROR MESSAGES.

input

KX - The maximum value of K. KX.LE.50

input

PP, RR - Arrays used for temporary storage in LEG2.

ERROR MESSAGES *** If the parameters M and NX are too large the program will be unable to continue execution and control will return to the label parameter L specified in the calling sequence after the following message has been printed:

*** ERROR IN LEG2 *** NUMBERS HAVE BECOME TOO LARGE

Also see ERROR MESSAGES in the description of LEG1.

ACCURACY *** In all cases tested there were D significant digits in the generated function values. However accuracy may be affected if X is very close to 1. The rate of convergence of this algorithm decreases as X increases.

```
      SUBROUTINE LEG2(X,A,M,NX,D,P2,K,*,KX,PAPRX,RR)
C *** THIS SUBROUTINE GENERATES, IN MULTIPLE PRECISION, TO D SIGNIFICANT
C *** DIGITS THE ASSOCIATED LEGENDRE FUNCTIONS OF THE FIRST KIND P(X),
C *** FOR INTEGER ORDER M AND DEGREE A,A+1,A+2,...,A+NX-1, A NOT AN
C *** INTEGER, AND X.GE.1
      COMMON /LONG/LN,LW,LT,LS,LR,LTR,LEX,LRM
      INTEGER LN,LW,LT,LS,LEX
      REAL LR,LTR,LRM
      INTEGER X(1),A(1),M,NX,D,K,KX,P2(KX,1),PAPRX(KX,1),RR(KX,1),
     1        TEMP(50),TEMP2(50),ONE(50)
      IF (M.LT.0)RETURN 1
      CALL LEG1(X,A,M+1,D,P2,K,&70,KX,PAPRX,RR)
      CALL COPY(P2(1,M+1),TEMP,K)
      IF (NX.GT.1)GO TO 10
      CALL COPY(TEMP,P2(1,1),K)
      RETURN
   10 CALL SET(ONE,1,&20,K)
   20 CALL ADD(A,ONE,TEMP2,K)
      CALL LEG1(X,TEMP2,M+1,D,P2,K,&70,KX,PAPRX,RR)
      CALL COPY(P2(1,M+1),P2(1,2),K)
      CALL COPY(TEMP,P2(1,1),K)
      IF (NX.EQ.2)RETURN
      NXM1 = NX-1
      DO 60 NN=2,NXM1
      N = NN-1
C
C *** P2(N+1)=((2*N+2*A+1)*X*P2(N)-(N+A+M)*P2(N-1))/(N+A-M+1),N=1,NMAX-1
      CALL SET(TEMP,2*N+1,&100,K)
      CALL ADD   (A,A,TEMP2,K)
      CALL ADD(TEMP,TEMP2,TEMP,K)
      CALL MLTPLY(TEMP,X,TEMP,K)
      CALL MLTPLY(TEMP,P2(1,NN),P2(1,NN+1),K)
      CALL SET(TEMP,N+M,&100,K)
      CALL ADD(TEMP,A,TEMP,K)
      CALL MLTPLY(TEMP,P2(1,NN-1),TEMP,K)
      CALL SBTRCT(P2(1,NN+1),TEMP,P2(1,NN+1),K)
      CALL SET(TEMP,N-M+1,&100,K)
      CALL ADD(TEMP,A,TEMP,K)
      CALL DIVIDE(P2(1,NN+1),TEMP,P2(1,NN+1),K)
   60 CONTINUE
      RETURN
   70 RETURN 1
C
C *** ERROR RETURN
  100 WRITE (6,1000)
 1000 FORMAT ('- *** ERROR IN LEG2 *** NUMBERS HAVE BECOME TOO LARGE')
      RETURN 1
      END
```

ENTRY NAME
CONIC

IDENTIFICATION *** Multiple Precision Conical Functions

PURPOSE *** This subroutine generates, in multiple precision, to D significant digits Mehler's conical functions P(X), for order N=0,1,...,NX-1 and degree -1/2+iτ and X.GE.1.

OTHER SUBROUTINES USED *** FIXPI, IBCOM, LNGCNS, SET, SBTRCT, COPY, MLTPLY, LSQRT, ADD, LNGLOG, LCOS, DIVIDE, MLTINT, SHORTN and RECIP.

USUAGE ***
```
INTEGER X(KX),TAU(KX),NX,D,P(KX,NXX),K,KX,PP(KX,NXX),RR(KX,NXX)
  :
CALL CONIC(X,TAU,NX,D,P,K,&L,KX,PP,RR)
```

where the parameters in the calling sequence are:

X - The multiple precision argument stored in the form X.EQ.A*(10**4)**B where 1.LE.A.LT.10**4 as follows:

X(1) = sign of the number: +1 if positive
-1 if negative
0 if zero (In this case, X(2) to X(K) are disregarded)

X(2) = exponent B of 10**4

X(3) = integral part of the mantissa, 1.LE.X(3).LT.10**4

X(4) to X(K) = fractional part of the mantissa, stored 4 decimal digits per location.

The value of X must be greater than or equal to one.

input

TAU - The conical functions will be generated for degree - 1/2+i*TAU, with TAU a multiple precision number stored in the same manner as X.

input

NX - The conical functions will be generated for order 0,1,...,NX-1.
NX.GT.0, NX.LE.NXX

input

D - The number of significant digits desired in the function values.
A suggested value to use is (K-5)*4+3.

input

P - On output, the Jth column, P(I,J), I=1,...,K will contain the multiple precision conical function of degree -1/2+i*TAU and order J-1 stored in the same manner as X.

output

K - The number of locations used in the arrays to store the multiple precision numbers. K.LE.KX.LE.50

input

L - Control returns immediately to the statement labelled L (an actual integer) if X.LT.1 or NX.LE.0. Also see ERROR MESSAGES.

input

KX - The maximum value of K. KX.LE.50

input

PP, RR - Arrays used for temporary storage in CONIC.

ERROR MESSAGES *** If the parameters TAU and NX are too large the program will be unable to continue execution and control will return to the label parameter L specified in the calling sequence after the following message has been printed:

*** ERROR IN CONIC *** NUMBERS HAVE BECOME TOO LARGE

Control is also returned to the label parameter L if the algorithm fails to converge due to D being set larger than can be attained. In this case the following message is printed:

*** CONICAL DOES NOT CONVERGE ***

ACCURACY *** In all cases tested there were D significant digits in the generated function values. This algorithm converges slowly when X and TAU are large.

```
      SUBROUTINE CONIC(X,TAU,NX,D,P,K,*,KX,PAPRX,RR)
C *** THIS SUBROUTINE GENERATES, IN MULTIPLE PRECISION, TO D SIGNIFICANT
C *** DIGITS MEHLER'S CONICAL FUNCTIONS P(X), FOR ORDER N=0,1,...,NX-1
C *** AND DEGREE (-1/2 + I*TAU) AND X.GE.1
      COMMON /LONG/LN,LW,LT,LS,LR,LTR,LEX,LRM
      INTEGER LN,LW,LT,LS,LEX
      REAL LR,LTR,LRM
      INTEGER X(1),TAU(1),NX,D,K,KX,P(KX,1),PAPRX(KX,1),RR(KX,1),N,NU,M,
     1        EPS(50),T(50),X1(50),X2(50),SUM(50),LMBDA1(50),LMBDA2(50),
     2        LMBDA(50),R(50),S(50),ZERO(50),ONE(50),TWO(50),TEMP(50),
     3        TEMP2(50),TEMP3(50),HALF(50)
      REAL XS,TAUS
      CALL LNGCNS(K,4,8,75)
C
C *** INITIALIZE CONSTANTS FOR LONG ARITHMETIC
      CALL SET(ZERO,0,&10,K)
   10 CALL SET(ONE,1,&20,K)
   20 CALL SET(TWO,2,&30,K)
   30 HALF(1) = 1
      HALF(2) = -1
      HALF(3) = 5000
      DO 35 I=4,K
   35 HALF(I) = 0
C
C *** CHECK IF INPUT PARAMETERS ARE IN CORRECT RANGE
      CALL SBTRCT(X,ONE,TEMP,K)
      IF (TEMP(1).LT.0.OR.NX.LE.0)RETURN 1
C
C *** IF X=1, P(0)=1, P(N)=0 N=1,NMAX
      IF (TEMP(1).GT.0)GO TO 50
      CALL COPY(ONE,P(1,1),K)
      IF (NX.EQ.1)RETURN
      DO 40 N=2,NX
      CALL COPY(ZERO,P(1,N),K)
   40 CONTINUE
      RETURN
C
C *** T = TAU**2
   50 CALL MLTPLY(TAU,TAU,T,K)
C
C *** PAPPROX(N)=0 FOR N=0,NMAX
      DO 60 N=1,NX
      CALL COPY(ZERO,PAPRX(1,N),K)
   60 CONTINUE
C
C *** EPSILON = .5*10**(-D)
      EPS(1) = +1
      EPS(2) = -D/4 - 1
      EPS(3) = 5 * 10**(-4*EPS(2) - D - 1)
      DO 70 I=4,K
   70 EPS(I) = 0
C
C *** X1 = SQRT(X**2-1)
      CALL MLTPLY(X,X,TEMP,K)
      CALL SBTRCT(TEMP,ONE,TEMP,K)
      CALL LSQRT(TEMP,X1,K)
C
```

```
C *** X2 = X + X1
      CALL ADD(X,X1,X2,K)
C
C *** SUM = COS(TAU*LN(X2))/SQRT(X2)
      CALL LNGLOG(X2,TEMP,&80,K)
   80 CALL MLTPLY(TEMP,TAU,TEMP,K)
      CALL LCOS(TEMP,TEMP,K)
      CALL LSQRT(X2,TEMP2,K)
      CALL DIVIDE(TEMP,TEMP2,SUM,K)
C
C *** X1 = 2*X/X1
      CALL MLTINT(X,2,TEMP,&90,K)
   90 CALL DIVIDE(TEMP,X1,X1,K)
C
C *** NU=30+ENTIER((1+(.140+.0246*TAU)*(X-1))*NMAX)
      CALL SHORTN(X,XS,&350,K)
      CALL SHORTN(TAU,TAUS,&350,K)
      NU = 30+AINT((1+(.140+.0246*TAUS)*(XS-1))*(NX-1))
      NUINC = 60
C
C *** L0: N=2
  110 N = 2
C
C *** LAMBDA1 = 1/(.25+T)
      TEMP(1) = 1
      TEMP(2) = -1
      TEMP(3) = 2500
      DO 120 I=4,K
  120 TEMP(I) = 0
      CALL ADD(TEMP,T,TEMP,K)
      CALL RECIP(TEMP,LMBDA1,K)
C
C *** LAMBDA2 = (3-4*T)/((.25+T)*(2.25+T))
      CALL SET(TEMP,3,&130,K)
  130 CALL MLTINT(T,4,TEMP2,&140,K)
  140 CALL SBTRCT(TEMP,TEMP2,LMBDA2,K)
      TEMP(1) = 1
      TEMP(2) = -1
      TEMP(3) = 2500
      DO 145 I=4,K
  145 TEMP(I) = 0
      CALL ADD(TEMP,T,TEMP,K)
      TEMP2(1) = 1
      TEMP2(2) = 0
      TEMP2(3) = 2
      TEMP2(4) = 2500
      DO 150 I=5,K
  150 TEMP2(I) = 0
      CALL ADD(TEMP2,T,TEMP2,K)
      CALL MLTPLY(TEMP,TEMP2,TEMP,K)
      CALL DIVIDE(LMBDA2,TEMP,LMBDA2,K)
C
C *** L1: LAMBDA = (1+1/N)*(2*LAMBDA2-LAMBDA1)/((1+.5/N)**2+(TAU/N)**2)
  160 CALL SET(TEMP,N,&500,K)
      CALL RECIP(TEMP,TEMP,K)
      CALL ADD(TEMP,ONE,TEMP2,K)
      CALL MLTINT(LMBDA2,2,LMBDA,&180,K)
```

```
  180 CALL SBTRCT(LMBDA,LMBDA1,LMBDA,K)
      CALL MLTPLY(LMBDA,TEMP2,LMBDA,K)
      CALL MLTPLY(TEMP,HALF,TEMP2,K)
      CALL ADD(ONE,TEMP2,TEMP2,K)
      CALL MLTPLY(TEMP2,TEMP2,TEMP2,K)
      CALL MLTPLY(TEMP,TAU,TEMP,K)
      CALL MLTPLY(TEMP,TEMP,TEMP,K)
      CALL ADD(TEMP2,TEMP,TEMP,K)
      CALL DIVIDE(LMBDA,TEMP,LMBDA,K)
C
C *** IF N.LT.NU THEN LAMBDA1=LAMBDA2; LAMBDA2=LAMBDA; N=N+1; GO TO L1
      IF (N.GE.NU)GO TO 200
      CALL COPY(LMBDA2,LMBDA1,K)
      CALL COPY(LMBDA,LMBDA2,K)
      N = N+1
      GO TO 160
C
C *** R = S = 0
  200 CALL COPY(ZERO,R,K)
      CALL COPY(ZERO,S,K)
C
C *** L2:R=-((1-.5/N)**2+(TAU/N)**2)/(X1+(1+1/N)*R)
  210 CALL SET(TEMP3,N,&500,K)
      CALL RECIP(TEMP3,TEMP3,K)
      CALL ADD(TEMP3,ONE,TEMP2,K)
      CALL MLTPLY(TEMP2,R,R,K)
      CALL ADD(R,X1,R,K)
      CALL MLTPLY(HALF,TEMP3,TEMP2,K)
      CALL SBTRCT(ONE,TEMP2,TEMP2,K)
      CALL MLTPLY(TEMP2,TEMP2,TEMP2,K)
      CALL MLTPLY(TEMP3,TAU,TEMP,K)
      CALL MLTPLY(TEMP,TEMP,TEMP,K)
      CALL ADD(TEMP2,TEMP,TEMP,K)
      CALL DIVIDE(TEMP,R,R,K)
      R(1) = -R(1)
C
C *** S = R * (LAMBDA2+S)
      CALL ADD(LMBDA2,S,S,K)
      CALL MLTPLY(S,R,S,K)
C
C *** IF N.LE.NMAX THEN RR(N-1)=R
      IF (N.GT.NX-1)GO TO 240
      CALL COPY(R,RR(1,N),K)
C
C *** LAMBDA1 = LAMBDA2
  240 CALL COPY(LMBDA2,LMBDA1,K)
C
C *** LAMBDA2=2*LAMBDA2-((1+.5/N)**2+(TAU/N)**2)*LAMBDA/(1+1/N)
      CALL MLTPLY(HALF,TEMP3,TEMP2,K)
      CALL ADD(ONE,TEMP2,TEMP2,K)
      CALL MLTPLY(TEMP2,TEMP2,TEMP2,K)
      CALL MLTPLY(TAU,TEMP3,TEMP,K)
      CALL MLTPLY(TEMP,TEMP,TEMP,K)
      CALL ADD(TEMP2,TEMP,TEMP,K)
      CALL MLTPLY(TEMP,LMBDA,TEMP,K)
      CALL ADD(ONE,TEMP3,TEMP3,K)
      CALL DIVIDE(TEMP,TEMP3,TEMP,K)
```

```
      CALL MLTINT(LMBDA2,2,LMBDA2,&270,K)
  270 CALL SBTRCT(LMBDA2,TEMP,LMBDA2,K)
C
C *** LAMBDA = LAMBDA1
      CALL COPY(LMBDA1,LMBDA,K)
C
C *** N=N-1; IF N.GE.1 THEN GO TO L2
      N = N - 1
      IF (N.GE.1)GO TO 210
C
C *** P(0) = SUM/(1+S)
      CALL ADD(ONE,S,TEMP,K)
      CALL DIVIDE(SUM,TEMP,P(1,1),K)
C
C *** FOR N=0,NMAX-1 DO P(N+1)=RR(N)*P(N)
      IF (NX.EQ.1)GO TO 290
      NXM1 = NX-1
      DO 280 N=1,NXM1
      CALL MLTPLY(RR(1,N),P(1,N),P(1,N+1),K)
  280 CONTINUE
  290 DO 300 N=1,NX
C
C *** IF (ABS(P(N)-PAPPROX(N)).GT.EPSILON*ABS(P(N))GO TO 320
      CALL SBTRCT(P(1,N),PAPRX(1,N),TEMP,K)
      TEMP(1) = 1
      CALL MLTPLY(EPS,P(1,N),TEMP2,K)
      TEMP2(1) = 1
      CALL SBTRCT(TEMP,TEMP2,TEMP,K)
      IF (TEMP(1).GT.0)GO TO 320
  300 CONTINUE
C
C *** T = 1
      CALL COPY(ONE,T,K)
C
C *** FOR N=1,NMAX DO T=N*T; P(N)=T*P(N)
      IF (NX.EQ.1)RETURN
      DO 310 NN=2,NX
      N = NN-1
      CALL MLTINT(T,N,T,&500,K)
  310 CALL MLTPLY(T,P(1,NN),P(1,NN),K)
      RETURN
C
C *** FOR M=0,NMAX DO PAPPROX(M)=P(M)
  320 DO 330 M=1,NX
      CALL COPY(P(1,M),PAPRX(1,M),K)
  330 CONTINUE
      NU = NU + NUINC
      NUINC = 2 * NUINC
      IF (NU.LT.5000)GO TO 110
      WRITE (6,1000)
 1000 FORMAT ('- *** CONICAL DOES NOT CONVERGE ***')
  350 RETURN 1
C
C *** ERROR RETURN
  500 WRITE (6,1001)
 1001 FORMAT ('- *** ERROR IN CONIC *** NUMBERS HAVE BECOME TOO LARGE')
      RETURN 1
      END
```

ENTRY NAME
TOROID

IDENTIFICATION *** Multiple Precision Toroidal Functions

PURPOSE *** This subroutine generates, in multiple precision, to D significant digits the toroidal functions of the second kind Q(X), with integer order M and degree - 1/2+N, N=0,1,...,NX-1 and X.GT.1.

OTHER SUBROUTINES USED *** FIXPI, IBCOM, LNGCNS, COPY, SET, SBTRCT, MLTPLY, ADD, DIVIDE, LSQRT, LEXPO and SHORTN.

USAGE *** INTEGER X(KX),M,NX,D,Q(KX,NXX),K,KX,QQ(KX,NXX),RR(KX,NXX)

```
      :
      :
      CALL TOROID(X,M,NX,D,Q,K,&L,KX,QQ,RR)
```

where the parameters in the calling sequence are:

X - The multiple precision argument stored in the form X.EQ.A.*(10**4)**B where 1.LE.A.LT.10**4 as follows:

X(1) = sign of the number: +1 if positive
-1 if negative
0 if zero (In this case, X(2) to X(K) are disregarded)

X(2) = exponent B of 10**4

X(3) = integral part of the mantissa, 1.LE.X(3).LT.10**4

X(4) to X(K) = fractional part of the mantissa, stored 4 decimal digits per location.

The value of X must be greater than one.

input

M - The integer order of the toroidal functions which will be generated.

input

NX - The toridal functions will be generated for degree - 1/2, 1/2,..., - 1/2+NX-1 NX.GT.0, NX.LE.NXX

input

D - The number of significant digits desired in the function values. A suggested value to use is (K-4)*4+1.

input

Q - On output, the Jth column, Q(I,J), I=1,...,K will contain the multiple precision toroidal function of order M and degree - 1/2 + J-1 stored in the same manner as X.

output

K - The number of locations used in the arrays to store the multiple precision numbers. K.LE.KX.LE.50

input

L - Control returns immediately to the statement labelled L (an actual integer) if X.LE.1 or NX.LE.0. Also see ERROR MESSAGES

input

KX - The maximum value of K. KX.LE.50

input

QQ, RR - Arrays used for temporary storage in TOROID.

ERROR MESSAGES *** In some cases the program will be unable to continue execution and control will return to the label parameter L specified in the calling sequence after the following message has been printed:

*** ERROR IN TOROID *** NUMBERS HAVE BECOME TOO LARGE

This can occur if M or NX are too large or if the number of significant digits D asked for it too great and the algorithm is failing to converge.

ACCURACY *** In all cases tested there were D significant digits generated

Convergence of the algorithm is slow for X near 1.

```
      SUBROUTINE TOROID(X,M,NX,D,Q,K,*,KX,QAPRX,RR)
C *** THIS SUBROUTINE GENERATES, IN MULTIPLE PRECISION, TO D SIGNIFICANT
C *** DIGITS THE TOROIDAL FUNCTIONS Q(X), WITH INTEGER ORDER M AND
C *** DEGREE (-1/2 + N), N=0,1,...,NX-1 AND X.GT.1
      COMMON /LONG/LN,LW,LT,LS,LR,LTR,LEX,LRM
      INTEGER LN,LN,LT,LS,LEX
      REAL LR,LTR,LRM
      INTEGER X(1),M,NX,D,K,KX,Q(KX,1),QAPRX(KX,1),RR(KX,1),N,NU,P,
     1        EPS(50),X1(50),C(50),SUM(50),R(50),S(50),ONE(50),TWO(50),
     2        TEMP(50),TEMP2(50),HALF(50),C1(50)
      REAL XS
C
C *** C = PI/SQRT(2)
      DATA    C1/   1,   0,   2,2214,4146,9079,1831,2350,7940,4950,3034,
     1 6849,3073,1084,4687,8451,1154,2697,8034,7821,7396,5497,3695,5287,
     2 6634,6738,2382,6186,8170,5106,3426,1439,5682,1797,9925,9172,8137,
     3 4618,7673,1798,3720,7511,6396,3940,9877,9387,6655,6726,8305,5065/
      CALL LNGCNS(K,4,8,75)
      CALL COPY(C1,C,K)
C
C *** INITIALIZE CONSTANTS FOR LONG ARITHMETIC
      CALL SET(ONE,1,&10,K)
   10 CALL SET(TWO,2,&20,K)
   20 HALF(1) = 1
      HALF(2) = -1
      HALF(3) = 5000
      DO 30 I=4,K
   30 HALF(I) = 0
C
C *** CHECK IF INPUT PARAMETERS ARE IN CORRECT RANGE
      CALL SBTRCT(X,ONE,TEMP,K)
      IF (TEMP(1).LE.0.OR.NX.LE.0)RETURN 1
C
C *** FOR N=0,NMAX DO QAPPROX(N)=0
      CALL SET(TEMP,0,&40,K)
   40 DO 50 N=1,NX
      CALL COPY(TEMP,QAPRX(1,N),K)
   50 CONTINUE
C
C *** EPSILON = .5 * 10**(-D)
      EPS(1) = 1
      EPS(2) = -D/4 - 1
      EPS(3) = 5 * 10**(-4*EPS(2) -D -1)
      DO 60 I=4,K
   60 EPS(I) = 0
C
C *** IF M.GE.0 THEN FOR N=0,M-1 DO C=-(N+.5)*C
      IF (M.LT.0)GO TO 95
      IF (M.EQ.0)GO TO 90
      CALL COPY(HALF,TEMP2,K)
      DO 80 NN=1,M
      CALL MLTPLY(TEMP2,C,C,K)
      C(1) = -C(1)
      CALL ADD(TEMP2,ONE,TEMP2,K)
   80 CONTINUE
   90 GO TO 120
C
```

```
C *** IF M.LT.0 FOR N=0,STEP -1,M+1 DO C = -C/(N-.5)
   95 CALL COPY(HALF,TEMP2,K)
      TEMP2(1) = -1
      MM = -M
      DO 110 NN=1,MM
      CALL DIVIDE(C,TEMP2,C,K)
      C(1) = -C(1)
      CALL SBTRCT(TEMP2,ONE,TEMP2,K)
  110 CONTINUE
C
C *** SUM = C*((X+1)/(X-1))**(M/2)/SQRT(X-1)
  120 CALL SBTRCT(X,ONE,TEMP,K)
      CALL ADD(X,ONE,TEMP2,K)
      CALL DIVIDE(TEMP2,TEMP,TEMP2,K)
      CALL LSQRT(TEMP,TEMP,K)
      CALL DIVIDE(C,TEMP,SUM,K)
      CALL SET(TEMP,M,&300,K)
      CALL MLTPLY(TEMP,HALF,TEMP,K)
      CALL LEXPO(TEMP2,TEMP,TEMP,K)
      CALL MLTPLY(SUM,TEMP,SUM,K)
C
C *** X1 = 2*X
      CALL ADD(X,X,X1,K)
C
C *** NU = 20 + ENTIER((1.15+(.0146+.00122*M)/(X-1))*NMAX)
      CALL SHORTN(X,XS,&350,K)
      NU = 20 +AINT((1.15+(.0146+.00122*M)/(XS-1))*(NX-1))
      NUINC = 10
C
C *** LO: R = S = 0
  140 CALL SET(TEMP,0,&150,K)
  150 CALL COPY(TEMP,R,K)
      CALL COPY(TEMP,S,K)
C
C *** FOR N=NU STEP -1 UNTIL 1
      DO 200 NN=1,NU
      N = NU-NN+1
C
C *** R = (N+M-.5)/(N*X1-(N-M+.5)*R)
      CALL SET(TEMP2,N-M,&300,K)
      CALL ADD(TEMP2,HALF,TEMP2,K)
      CALL MLTPLY(TEMP2,R,R,K)
      CALL SET(TEMP2,N,&300,K)
      CALL MLTPLY(TEMP2,X1,TEMP2,K)
      CALL SBTRCT(TEMP2,R,R,K)
      CALL SET(TEMP2,N+M,&300,K)
      CALL SBTRCT(TEMP2,HALF,TEMP2,K)
      CALL DIVIDE(TEMP2,R,R,K)
C
C *** S = R * (2+S)
      CALL ADD(TWO,S,S,K)
      CALL MLTPLY(R,S,S,K)
C
C *** IF N.LE.NMAX THEN RR(N-1) = R
      IF (N.GE.NX)GO TO 200
      CALL COPY(R,RR(1,N),K)
  200 CONTINUE
```

```
C
C *** Q(0) = SUM/(1+S)
      CALL ADD(ONE,S,TEMP,K)
      CALL DIVIDE(SUM,TEMP,Q(1,1),K)
C
C *** FOR N=0,NMAX-1 DO Q(N+1) = RR(N) * Q(N)
      IF (NX.EQ.1)GO TO 220
      NXM1 = NX-1
      DO 210 N=1,NXM1
      CALL MLTPLY(RR(1,N),Q(1,N),Q(1,N+1),K)
  210 CONTINUE
  220 DO 230 N=1,NX
C
C *** IF (ABS(Q(N)-QAPPROX(N)).GT.EPSILON*ABS(Q(N))GO TO 240
      CALL SBTRCT(Q(1,N),QAPRX(1,N),TEMP,K)
      TEMP(1) = +1
      CALL MLTPLY(EPS,Q(1,N),TEMP2,K)
      TEMP2(1) = +1
      CALL SBTRCT(TEMP,TEMP2,TEMP,K)
      IF (TEMP(1).GT.0)GO TO 240
  230 CONTINUE
      RETURN
C
C *** FOR P=0,NMAX DO QAPPROX(P)=Q(P)
  240 DO 250 P=1,NX
      CALL COPY(Q(1,P),QAPRX(1,P),K)
  250 CONTINUE
C
C *** NU = NU+10; GO TO LO
      NU = NU + NUINC
      NUINC = 2 * NUINC
      GO TO 140
C
C *** ERROR RETURN
  300 WRITE (6,1000)
 1000 FORMAT ('- *** ERROR IN TOROID *** NUMBERS HAVE BECOME TOO LARGE')
  350 RETURN 1
      END
```

AEC–427
5/68)
M 3201

U. S. ATOMIC ENERGY COMMISSION

UNIVERSITY–TYPE CONTRACTOR'S RECOMMENDATION FOR DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

EC REPORT NO.
COO-1469-0167

2. TITLE
MULTIPLE PRECISION COMPUTATION OF LEGENDRE FUNCTIONS

YPE OF DOCUMENT (Check one):

- [X] a. Scientific and technical report
- [ ] b. Conference paper not to be published in a journal:
  - Title of conference _____
  - Date of conference _____
  - Exact location of conference _____
  - Sponsoring organization _____
- [ ] c. Other (Specify) _____

ECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

- [X] a. AEC's normal announcement and distribution procedures may be followed.
- [ ] b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.
- [ ] c. Make no announcement or distrubution.

EASON FOR RECOMMENDED RESTRICTIONS:

SUBMITTED BY: NAME AND POSITION (Please print or type)

C. W. Gear, Professor
and Principle Investigator

Organization

Signature Charles W Gear.

Date June 1970

FOR AEC USE ONLY

AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

PATENT CLEARANCE:

- [ ] a. AEC patent clearance has been granted by responsible AEC patent group.
- [ ] b. Report has been sent to responsible AEC patent group for clearance.
- [ ] c. Patent clearance not required.